CU-CAS-96-29

CENTER FOR AEROSPACE STRUCTURES

HIGH-PERFORMANCE PARALLEL ANALYSIS OF COUPLED PROBLEMS FOR AIRCRAFT PROPULSION

Final Progress Report to NASA LeRC on Grant NAG3-1425

by

C. A. Felippa, C. Farhat, K. C. Park, U. Gumaste,
P.-S. Chen and M. Lesoinne and P. Stern

December 1996

COLLEGE OF ENGINEERING
UNIVERSITY OF COLORADO
CAMPUS BOX 429
BOULDER, COLORADO 80309

TABLE OF CONTENTS

SUMMARY

This research program dealt with the application of high-performance computing methods to the numerical simulation of complete jet engines. The program was initiated in January 1993 by applying two-dimensional parallel aeroelastic codes to the interior gas flow problem of a by-pass jet engine. The fluid mesh generation, domain decomposition and solution capabilities were successfully tested. Attention was then focused on methodology for the partitioned analysis of the interaction of the gas flow with a flexible structure and with the fluid mesh motion driven by these structural displacements. The latter is treated by a ALE technique that models the fluid mesh motion as that of a fictitious mechanical network laid along the edges of near-field fluid elements. New partitioned analysis procedures to treat this coupled three-component problem were developed during 1994 and 1995. These procedures involved delayed corrections and subcycling, and have been successfully tested on several massively parallel computers, including the iPSC-860, Paragon XP/S and the IBM SP2. For the global steady-state axisymmetric analysis of a complete engine we have decided to use the NASA-sponsored `ENG10` program, which uses a regular FV-multiblock-grid discretization in conjunction with circumferential averaging to include effects of blade forces, loss, combustor heat addition, blockage, bleeds and convective mixing. A load-balancing preprocessor for parallel versions of `ENG10` was developed. During 1995 and 1996 we developed the capability for the first full 3D aeroelastic simulation of a multirow engine stage. This capability was tested on the IBM SP2 parallel supercomputer at NASA Ames. Benchmark results were presented at the 1196 Computational Aeroscience meeting.

1. OVERVIEW

The present program deals with the application of high-performance parallel computation for the analysis of complete jet engines, considering the interaction of fluid, thermal and mechanical components. The research is driven by the simulation of advanced aircraft propulsion systems, which is a problem of primary interest to NASA Lewis.

The coupled problem involves interaction of structures with gas dynamics, heat conduction and heat transfer in aircraft engines. The *methodology* issues to be addressed include: consistent discrete formulation of coupled problems with emphasis on coupling phenomena; effect of partitioning strategies, augmentation and temporal solution procedures; sensitivity of response to problem parameters; and methods for interfacing multiscale discretizations. The *computer implementation* issues to be addressed include: parallel treatment of coupled systems; domain decomposition and mesh partitioning strategies; data representation in object-oriented form and mapping to hardware driven representation, and tradeoff studies between partitioning schemes with differing degree of coupling.

2. STAFF

The present grant began in January 1993 and concluded in July 1996. Two graduate students were supported during that period. M. Ronaghi (a U.S. citizen) began his graduate studies at the University of Colorado on January 1993. He completed a M.Sc. in Aerospace Engineering on May 1994 and left to join Analytics Inc. (Hampton, VA) on June 1994.

U. Gumaste (a permanent U.S. resident) began his graduate studies at Colorado in August 1993. Mr. Gumaste received a B.Tech in Civil Engineering from the Indian Institute of Technology, Bombay, India. He completed his Ph. D. course requirement in the Fall 1994 semester with the transfer of graduate credit units from the University of Maryland. On November 1994 and December 1996 he passed the Preliminary Exam and Comprehensive Exam, respectively, for the Ph. D. degree, and is scheduled to complete his Ph. D. by the end of 1997. He became familiar with our external aeroelastic codes starting in the summer of 1994. He visited NASA Lewis for five weeks during July–August 1994, and for four weeks during June-July 1995.

One Post-Doctoral Research Associate, Paul Stern, was partly supported by this grant during 1994-95 for running benchmarks on parallel computers. One graduate student, P.-S. Chen, supported by a related grant from NASA Ames, assisted with model preparation tasks over the period 1995-96.

The development of FSI methodology for this project has benefited from the presence of several Visiting Scholars whose work concentrated on the related problem of exterior aeroelasticity for a complete aircraft. This project was part of a Grant Challenge Applications Award supported by NSF, but most aspects of the solution methodology and parallel implementation are applicable to FSI engine problems. Dr. S. Lanteri conducted extensive experimentation on several computational algorithms for compressive viscous flow simulation on the iPSC-860, CM-5 and KSR-1 as reported in the July 1994 Progress Report. Dr. N. Maman implemented "mesh matching" techniques that connect separately generated fluid and structural meshes. Dr. S. Piperno developed and evaluated

implicit and subcycled partitioned analysis procedures for the interaction of structure, fluid and fluid-mesh motion. A new approach to augmentation of the governing semi-discrete equations that improves stability while keeping communications overhead modest was investigated. Finally, Dr. M. Lesoinne (who finished his Ph.D. under C. Farhat on August 1994 and is presently a Research Faculty) made significant contributions to the modeling and computational treatment of the fluid mesh motion, and to the development of global conservation laws that must be obeyed by those motions.

Results from these studies are collected in a series of reports and papers. The major ones are enclosed as Appendices to the present report.

3. DEVELOPMENT OF PARTITIONED ANALYSIS METHODS

The first parallel computations of a jet engine, presented in the first progress report of July 1993 and reporoduced here as Appendix I, dealt with the fluid flow within a jet engine structure that is considered rigid and hence provides only guiding boundary conditions for the gas flow. When the structural flexibility is accounted for two complications occur:

1. The engine simulation algorithm must account for the structural flexibility though periodic transfer of interaction information, and
2. The fluid mesh must smoothly follow the relative structural motions through an ALE (Adaptive Lagrangian Eulerian) scheme. The particular ALE scheme selected for the present work makes use of Batina's proposed pseudo-mechanical model of springs and masses overlaid over the fluid mesh.

Research work during the period July 1993 through July 1996 was dominated by the treatment of two subjects: partitioned analysis of fluid-structure interaction (FSI) and accounting for fluid mesh motions. The partitioned analysis algorithm developed for the FSI problem is always implicit in the structure (because of its larger time scale of significant vibratory motions) and either explicit or implicit for the gas flow modeled by the Navier-Stokes equations. Subcycling, in which the integration stepsize for the fluid may be smaller than that used in the structure, was also studied.

3.1. General Requirements

The fundamental practical considerations in the development of these methods are: (1) numerical stability, (2) fidelity to physics, (3) accuracy, and (4) MPP efficiency. Numerical stability is fundamental in that an unstable method, no matter how efficient, is useless. There are additional considerations:

1. Stability degradation with respect to that achievable for the *uncoupled* fields should be minimized. For example, if the treatment is implicit-implicit (I-I) we would like to maintain unconditional stability. If the fluid is treated explicitly we would like to maintain the same CFL stability limit.
2. Masking of physical instability should be avoided. This is important in that flutter or divergence phenomena should not be concealed by numerical dissipation. For this reasons all time integration algorithms considered in this work must exclude the use of artificial damping.

3.2. Stability vs. Communication-Overhead Tradeoff

The degradation of numerical stability degradation is primarily influenced by the nature of information exchanged every time step among the coupled subsystems during the course of partitioned integration. A methodology called *augmentation* that systematically exploits this idea was developed by Park and Felippa in the late 1970s. The idea is to modify the governing equations of one subsystem with system information from connected subsystems. The idea proved highly successful for the sequential computers of the time. A fresh look must be taken to augmentation, however, in light of the communications overhead incurred in massively parallel processing. For the present application three possibilities were considered:

No augmentation. The 3 subsystems (fluid, structure and ALE mesh) exchange only minimal interaction state information such as pressures and surface-motion velocities, but no information on system characteristics such as mass or stiffness. The resulting algorithm has minimal MPP communication overhead but poor stability characteristics. In fact the stability of an implicit-implicit scheme becomes conditional and not too different from that of a less expensive implicit-explicit scheme. This degradation in turn can significantly limit the stepsize for both fluid and structure.

Full augmentation. This involves transmission of inverse-matrix-type data from one system to another. Such data are typified by terms such as a a structure-to-fluid coupling-matrix times the inverse of the structural mass. Stability degradation can be reduced or entirely eliminated; for example implicit-implicit unconditional stability may be maintained. But because the transmitted matrix combinations tend to be much less sparse than the original system matrices, the MPP communications overhead can become overwhelming, thus negating the benefits of improved stability characteristics.

Partial augmentation. This new approach involves the transmission of coupling matrix information which does not involve inverses. It is efficiently implemented as a delayed correction to the integration algorithm by terms proportional to the squared stepsize. The MPP communication requirements are modest in comparison to the fully-augmented case, whereas stability degradation can be again eliminated with some additional care.

The partial augmentation scheme was jointly developed by S. Piperno and C. Farhat in early 1994. Its derivation was reported in the July 1994 report and is enclosed here as Appendix II.

The use of these methods in three-dimensional aeroelasticity has been investigated from the summer 1994 to the present time. This investigation has resulted in the development of four specific algorithms for explicit/implicit staggered time integration, which are labeled as A0 through A4. The basic algorithm A0 is suitable for sequential computers when the time scale and computational cost of fluid and structure components is comparable. Algorithm A1 incorporates fluid subcycling. Algorithms A2 and A3 aim to exploit inter-field parallelism by allowing the integration over fluid and structure to proceed concurrently, with A3 aimed at achieving better accuracy through a more complex field synchronization scheme. These algorithms are described in more detail in Appendix II of this report.

3.3. Effects of Moving Fluid Mesh

The first one-dimensional results on the effect of a dynamic fluid mesh on the stability and accuracy of the staggered integration were obtained by C. Farhat and S. Piperno in late 1993 and early 1994, and are discussed in Appendix II of the July 1994 report. A doctoral student, M. Lesoinne (presently a post-doctoral Research Associate supported by a related NSF grant) extended those calculations to the multidimensional case. This work culminated in the development of a geometric conservation law (GCL) that must be verified by the mesh motion in the three-dimensional case. This law applies to unstructured meshes typical of finite element and finite-volume fluid discretizations, and extends the GCL enunciated for regular finite-difference discretizations by Thomas and Lombard in 1977. This new result is presented in more detail in Appendix II of this report.

3.4. Benchmarking on Parallel Computers

The new staggered solution algorithms for FSI, in conjunction with the improved treatment of fluid mesh motion dictated by the GCL, have been tested on several massively-parallel computational platforms using benchmark aerodynamic and FSI problems. These platforms include the Intel i860 Hypercube, Intel XP/S Paragon, Cray T3D, and IBM SP2. Performance results from these tests are reported and discussed in Appendix I of this report.

4. MODELING OF COMPLETE ENGINE

Work on the global model of a complete engine proceeded through two phases during 1994.

4.1. Homogenized Modeling of Compressor and Combustion Areas

Initial work in this topic in the first six months of 1994 was carried out using "energy injection" ideas. This idea contemplated feeding (or removing) kinetic and thermal energy into fluid mesh volume elements using the total-energy variables as "volume forcing" functions.

Although promising, energy injection in selected blocks of fluid volumes was found to cause significant numerical stability difficulties in the transient gas-flow analysis, which used explicit time integration. Consequently the development of these methods was put on hold because of the decision to use the program `ENG10` (which is briefly described in 4.2 below) for flow global analysis. `ENG10` makes use of similar ideas but the formulation of the governing equations and source terms in a rotating coordinate system is different. In addition a semi-implicit multigrid method, rather than explicit integration, is used to drive the gas flow solution to the steady state condition, resulting in better stability characteristics.

4.2. Load Balancing Preprocessor for Program `ENG10`

As a result of Mr. Gumaste's visit to NASA Lewis during July-August of 1994, it was decided to focus on the `ENG10` code written by Dr. Mark Stewart of NYMA Research Corp. to carry out the parallel analysis of a complete engine. This program was developed under contract with NASA Lewis, the contract monitors of this project being Austin Evans and Russell Claus.

ENG10 is a research program designed to carry out a "2 1/2" dimensional" flow analysis of a complete turbofan engine taking into account — through appropriate circumferential averaging — blade forces, loss, combustor heat addition, blockage, bleeds and convective mixing. The engine equations are derived from the three-dimensional fluid flow equations in a rotating cylindrical coordinate system. The Euler fluid flow equations express conservation of mass, momentum and rothalpy. These equations are discretized by structured finite-volume (FV) methods. The resulting discrete model is treated by multiblock-multigrid solution techniques. A multiblock grid divides the computational domain into topologically rectangular blocks in each of which the grid is regular (structured). For bladed jet engine geometries, this division is achieved through a series of supporting programs, namely TOPOS, TF and MS.

During the period September through December 1994, Mr. Gumaste devoted his time to the following tasks.

(a) Understanding the inner workings of ENG10 and learning to prepare inputs to this program (for which there is no user manual documentation) with the assistance from Dr. Stewart.

(b) Provide for links to the pre/postprocessor TOPS/DOMDEC developed by Charbel Farhat's group to view the decomposed model and analysis results.

(c) Develop and test algorithms for load-balancing the aerodynamic analysis of ENG10 in anticipation of running that program on parallel computers. The algorithm involves iterative merging and splitting of original blocks while respecting grid regularity constraints. This development resulted in a Load-Balancing (LB) program that can be used to adjust the original multiblock-grid discretization before starting ENG10 analysis runs on remote parallel computers (or local workstation networks).

Subtasks (b) and (c) were tested on a General Electric Energy Efficient Engine (GE-EEE) model provided by Dr. Stewart. A report on the development of LB is provided in Appendix III of this report.

5. 3D AEROELASTIC SIMULATIONS OF ENGINE ROWS

The final year of the grant were devoted to the aeroelastic simulation of multiple rows of the compressor stage of the GE EEE engine. Progress in that activity is summarized here.

5.1 The Aeroelastic Program PARFSI

This program treats the coupled aeroelastic problem following the partitioned analysis outlined previously. This strategy allows the development of tailored methods for each discipline component independently of the others. Also, new physical or computational partitions can be added to existing systems without substantial modifications to software modules that have attained stability. The main software components of PARFSI are briefly outlined below.

Fluid Solver. An Eulerian, explicit 3D Navier-Stokes solver based on Van Leer's Monotic Upwind System Conservation Laws (MUSCL) scheme [8]. May be reduced to an Euler solver for cases where viscosity effects are secondary, with a substantial (over 10 fold) speed gain. The convective

flux is handled by a finite volume discretization while a Galerkin finite element discretization is used for the diffusive flux. Non-overlapping domain decomposition is used for parallelization. The MIMD implementation of the code has been extensively tested on the iPSC-860, KSR-1 and Paragon. Preliminary results on the IBM SP-2 at NASA Ames were obtained during 1994 [4] with initial production results on multiblade configurations described in Subsection 3.3. The fluid code also runs efficiently on shared memory supercomputers such as the Cray C90 and YMP, and on workstation networks.

Structure Solver: A Lagrangian, implicit structure integrator based on the FETI (Finite Element Tearing and Interconnecting) mesh decomposition method. Mesh subdomains are condensed to the boundary by a direct solver. The interface problem is solved for Lagrange multiplier interpolants using projected/preconditioned conjugate gradients. In dynamic analysis, performance is further enhanced by a convergence accelerator that "remembers" the set of conjugate directions at the previous step. This solver has exhibited excellent MPP scalability [1–3].

Domain Decomposer: The pre- and post-processor program `TOP-DOMDEC` [2] has been developed for domain decomposition and dynamic visualization. This program performs automatic domain decomposition of fluid and structure discretizations and submits simulation runs to remote supercomputers. This program operates on SGI and IBM workstations using the GL graphics library and has a state-of-the-art "point and click" user-interface. In addition to `TOP-DOMDEC` a load balancing program for multiblock grids was developed [4] for the `ENG10` program described above.

Mesh Transfer. As noted previously the structure and fluid meshes are independently constructed and thus generally do not conform on the fluid-structure interfaces. The `MATCHER` preprocessor program [7] handles the initial process of information transfer between coupled but mismatched discretizations. This program uses a consistent surface interpolation approach and prepares the necessary decomposition so that interface data transfers can occur in parallel during the time-integration simulation.

Near-Field Fluid Mesh Motion. The ALE-mesh partition is handled through a spring-mass-dashpot network of fictitious mechanical elements placed over edges of the near-field fluid elements [5,6]. This network is implicitly time-integrated by the same techniques used in the structural solver.

5.2 Parallel Analysis of a Multi-Fan-Blade Configuration

The first three-dimensional aeroelastic analysis involving a multiple fan-blade configuration was successfully performed during October 1995 using `PARFSI` on the NAS/IBM SP2 at NASA Ames. This massively parallel supercomputer has 144 processing nodes (being expanded to 200+ as of this writing). Its nominal aggregate peak speed is over 100 Gigaflops, which puts it among the class of the most powerful MPP platforms worldwide.

Resources for this simulation were provided as part of a resource competition solicited by the CAS Office at NASA Ames in support of ongoing or new HPCC projects of relevance to NASA. An 8000-hour SP2 account for the Operational Year 95-96 was awarded on September 1995 and enabled by October 1st. This award was important in expediting these large simulations because the latest version of `PARFSI` simulation modules, which contain new capabilities relevant to the

engine problem, was developed on the IBM SP2, and makes extensive use of the message-passing protocol MPI just provided by IBM for this system.

The aeroelastic model used for the simulation presented here comprises one half of a blade row that pertains to the compression stage of a GE EEE turbofan engine. This reduced but realistic configuration was used to test the fluid and structure mesh generators, mesh matchers and analysis modules. This test model has approximately 185,000 degrees of freedom. This simulation is a prelude to the treatment of more complex configurations involving two to four full-circle blade rows. Such models are expected to contain up to 1.5 million freedoms, which is close to the computational limit on present massively parallel computing platforms such as the IBM SP2 and the Cray T3E.

The elastic structure contains 17 turbine blades attached to a fixed hub. The finite element model was directly generated, through step rotations, from a single GE EEE fan-stage blade NASTRAN model provided by Scott Thorpe of NASA Lewis Research Center. A very coarse model using triangular shell elements with "drilling" rotational degrees of freedom Each blade has 50 nodes, 72 triangular elements and 270 degrees of freedom. The structural mesh is half of that shown in Figure IV.1 of Appendix IV. For parallel analysis the structural mesh was kept as a single subdomain because of its low number of total degrees of freedom.

The fluid mesh was constructed in three steps. Following advice from David Miller of NASA LeRC, S-interpolation between two adjacent blades surfaces was used to generate a regular hexahedral mesh. Each hexahedron was then divided into six tetrahedra as expected by the PARFSI fluid solver. This mesh unit was step-rotated around the hub to fill the 16 spaces between the 17 blades. The full mesh was translated forward and backward to generate two inter-row transition volumes. The fluid mesh is half of that shown in Figure IV.1 of Appendix IV. The mesh contains approximately 185,000 defrees of freedom. For parallel processing a decomposition into 16 subdomains was performed by the TOP/DOMDEC preprocessor.

A uniform longitudinal flow of 0.8M is applied to the nodes of the fluid mesh. It is left to runs through the rigid blades until a steady state is reached. Then the blades are released except for the end ones which are maintained fixed. The blades are set into motion by the transverse forces induced by their skew angles, and vibrate approximately in phase. The total physical simulation time was 20 seconds, with 400 times steps performed in the structure and 8,000 steps on the fluid. Elapsed simulation time, using 28 processors of the NAS IBM SP2, was approximately 20 minutes. A color videotape of the dynamic response was prepared using the TOP/DOMDEC visualization system and provided to NASA Lewis.

5.3 Parallel Analysis of a Full Circle, Multiple-Row Configuration

Our final engine model involves a full circle of compressor blades as well as one and two-row configuration. This work is described in detail in Appendices IV and V.

6. FUTURE RESEARCH AREAS

In our opinion, the following research areas represents a natural continuation of the work funded under the present grant. The tasks outlined below represent a balanced combination among analysis

of more complex and demanding multirow configurations, improvements in the physics of the coupled model, ability to receive ENG10 inputs and compare three-dimensional stage results to those of the ENG10 axisymmetric idealization.

6.1 Multirow Fan Blade Simulations

It would be desirable to continue the aeroelastic simulations initiated with the model described in Section 5 fan stage until achieving the practical limits of the IBM SP2 and Cray T3D. Euler fluid models will be generally used to speed up the simulations, but Navier-Stokes models may be occasionally run to check the formation of shocks especially in unsteady conditions.

These models may be used as focus problems to explore the advantages of the present approach as well as to assess limits imposed by practical availability of computer resources such as processing power, physical memory and archival storage and communication bandwidth to move data from remote supercomputer sites to the visualization laboratory.

6.2 Unsteady Flow Analysis

While PARFSI is intrinsically designed to provide time-accurate unsteady analysis, its original development for the exterior aeroelastic flutter problem constrained the ability to provide time-dependent boundary conditions on the exterior fluid boundaries. This work would provide the ability to step up or decrease the engine inlet flow from one operating condition to another (or to an emergency condition) and conduct to drive the unsteady analysis.

6.3 Differential Rotation

Presently PARFSI assumed that the fluid mesh is Eulerian but inertial. This task provides the ability to model correctly the engine rotation by letting the fluid mesh rotate as a rigid body at a given speed. This speed may change during the course of the analysis if a time-accurate unsteady capability is incorporated.

Provision of this capability requires two modeling enhancements in the fluid model: (1) rotation induced source terms in the fluid, and (2) accounting for the gap between the rotating blades and the inertially fixed case with an attached non-rotating fluid mesh. The latter is truly a leading-edge research item that has not been previously considered to this level of modeling detail.

6.4 Geometric Stiffness Effects

Blade rotation produces a high tension stresses in the blades, which in turn affects their effective stiffness through the geometric stiffness matrix of the shell elements used in the finite element discretization. This capability would provide the necessary rotation-speed-to-stress feedback in the structural analyzer.

6.5 Other Coupling Effects

Coupling of structural material properties with the thermal solution provided by ENG10 may be considered during if interaction of thermal and aeroelastic effects are deemed important. Such effects may be of interest for blades fabricated with advanced composite materials.

7. REFERENCES

1. C. Farhat, L. Crivelli and F. X. Roux, A transient FETI methodology for large-scale parallel implicit computations in structural mechanics, *Internat. J. Numer. Meths. Engrg.*, **37**, (1994) 1945–1975.

2. C. Farhat, S. Lanteri and H. D. Simon, TOP/DOMDEC, A software tool for mesh partitioning and parallel processing, *J. Comput. Sys. Engrg.*, in press.

3. C. Farhat, P. S. Chen and P. Stern, Towards the ultimate iterative substructuring method: combined numerical and parallel scalability, and multiple load cases, *J. Comput. Sys. Engrg.*, in press.

4. C. A. Felippa, C. Farhat, P.-S. Chen, U. Gumaste, M. Lesoinne and P. Stern, High performance parallel analysis of coupled problems for aircraft propulsion, Progress Report to NASA LeRC for Period 6/94 through 1/95, Report CU-CAS-95-02, Center for Aerospace Structures, University of Colorado, Boulder, February 1995.

5. M. Lesoinne and C. Farhat, Stability analysis of dynamic meshes for transient aeroelastic computations, AIAA Paper No. 93-3325, *11th AIAA Computational Fluid Dynamics Conference*, Orlando, Florida, July 6–9, 1993.

6. M. Lesoinne and C. Farhat, Geometric conservation laws for aeroelastic computations Using unstructured dynamic meshes, AIAA Paper No. 95-1709, 1995.

7. N. Maman and C. Farhat, Matching fluid and structure meshes for aeroelastic computations: a parallel approach, *Computers & Structures*, in press

8. B. Van Leer, Towards the ultimate conservative difference scheme V: a second-order sequel to Godunov's method, *J. Comp. Phys.*, **32** (1979).

Appendix I
Theoretical Background on Viscous Flow Computations

Summary

The following material, extracted from a recently published paper by Farhat, Fezoui and Lanteri [3], summarizes the theoretical foundations of our parallel Navier-Stokes computations on unstructured meshes. Although the article focuses on CM-2 computations carried out during 1990-1991, it also presents implementation considerations applicable to the present project.

I.1 Introduction

Previously we have reported on our experience with performing two-dimensional structured compressible flow computations on the Connection Machine CM-2 (Saati, Biringen and Farhat [1], Lanteri, Farhat and Fezoui [2]). We have found that this massively parallel processor is particularly well suited for explicit computations on regular grids. For grids that result in a high virtual processor ratio (VPR or VP ratio), using the NEWS fast communication mechanism, we have measured the communication component of the simulation time to represent typically less than 10% of the total CPU time. We have concluded that on a 64K machine (65536 processors), efficiency rates in the neighborhood of 2 gigaflops are attainable. We have also found that for both inviscid (Euler equations) and viscous (Navier-Stokes equations) flow structured computations, a 16K CM-2 (16384 processors) can be 4 and 6 times faster than one CRAY-2 processor, respectively.

We focus here on massively parallel viscous flow computations using fully unstructured grids. In Section 2, we formulate the problem to be solved, and in Section 3, we derive first-order and second-order spatial schemes that are characterized by an upwind integration of the convective fluxes. Second-order accuracy is achieved through a Monotonic Upwind Scheme for Conservation Laws (MUSCL) technique. An explicit, and therefore nicely parallelizable, Runge-Kutta method is selected for time integration; it is summarized in Section 4. Because the mesh irregularities inhibit the use of the NEWS mechanism, interprocessor communication is bound to be carried out via the slower machine router. If a trivial processor mapping is used, up to 60% of the total CPU time is consumed in communication requirements. This bottleneck has been previously analyzed and documented by Farhat, Sobh and Park [3] for massively parallel finite element computations in solid mechanics problems. It has also been recently addressed by several other investigators for fluid flow computations. In particular, Shapiro [4] has proposed the use of a graph coloring algorithm to allow a particular implementation of the communication steps which reduces the communication costs by a factor of two. Hammond and Barth [5] have developed a vertex-based partitioning scheme for inviscid flow computations which attempts to minimize both the computational and communication costs associated with unstructured grids. Here, we present a strategy for mapping thousands of processors onto an unstructured grid which leads to an efficient scheme for carrying out communications of an arbitrary pattern. The key elements of this strategy are discussed in Section 5. These include the selection of an appropriate parallel data structure, the partitioning of a given unstructured grid into subgrids, and the mapping of each individual processor onto an entity

of these subgrids. Combining this mapping strategy with a communication compiler reduces the communication overhead by an order of magnitude and brings it down to 15% of the total simulation time. In Section 6, we apply our massively parallel code and its highly vectorized variant to the simulation of low Reynolds number chaotic flows. Measured performance results indicate that for such computations on unstructured grids, an 8K CM-2 with single precision floating point hardware is as fast as one CRAY-2 processor.

I.2. Mathematical modeling

First we recall the mathematical problem to be solved, and introduce the notation that is used in the sequel.

I.2.1. Governing equations

Let $\Omega \subset \Re^2$ be the flow domain of interest and Γ be its boundary. The conservative law form of the equations describing two-dimensional Navier-Stokes flows is given by :

$$\frac{\partial W}{\partial t} + \vec{\nabla} . \vec{\mathcal{F}}(W) = \frac{1}{Re} \vec{\nabla} . \vec{\mathcal{R}}(W) \tag{1}$$

where

$$\begin{aligned} W &= (\rho,\ \rho u,\ \rho v,\ E)^T \\ \vec{\nabla} &= \left(\frac{\partial}{\partial x},\ \frac{\partial}{\partial y} \right)^T \\ \vec{\mathcal{F}}(W) &= \begin{pmatrix} F(W) \\ G(W) \end{pmatrix} \\ \vec{\mathcal{R}}(W) &= \begin{pmatrix} R(W) \\ S(W) \end{pmatrix} \end{aligned} \tag{2}$$

The functions $F(W)$ and $G(W)$, and $R(W)$ and $S(W)$, denote the convective and diffusive fluxes, respectively. They can be written as :

$$
\begin{aligned}
F(W) &= \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix} \\
G(W) &= \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix} \\
R(W) &= \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} + \frac{\gamma k}{Pr}\frac{\partial \varepsilon}{\partial x} \end{pmatrix} \\
S(W) &= \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{xy} + v\tau_{yy} + \frac{\gamma k}{Pr}\frac{\partial \varepsilon}{\partial y} \end{pmatrix}
\end{aligned}
\tag{3}
$$

where ρ is the density, $\vec{U} = (u, v)$ is the velocity vector, E is the total energy per unit of volume, p is the pressure, and ε is the specific internal energy. The variables p, E, ρ, $\vec{U}$, ε, and the temperature T are related by the state equation for a perfect gas:

$$
p = (\gamma - 1)(E - \frac{1}{2}\rho \| \vec{U} \|^2) \tag{4}
$$

and by:

$$
\varepsilon = C_v T = \frac{E}{\rho} - \frac{1}{2}(\| \vec{U} \|^2) \tag{5}
$$

where γ denotes the ratio of specific heats.

The components of the Cauchy stress tensor τ_{xx}, τ_{xy} and τ_{yy} are given by:

$$
\tau_{xx} = \frac{2}{3}\mu\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) \quad \tau_{yy} = \frac{2}{3}\mu\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}\right) \quad \tau_{xy} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{6}
$$

where μ and k are the normalized viscosity and thermal conductivity coefficients. Two characteristic numbers appear in the above equations; the Reynolds number $Re = \dfrac{\rho_0 U_0 L_0}{\mu_0}$ where ρ_0, U_0, L_0

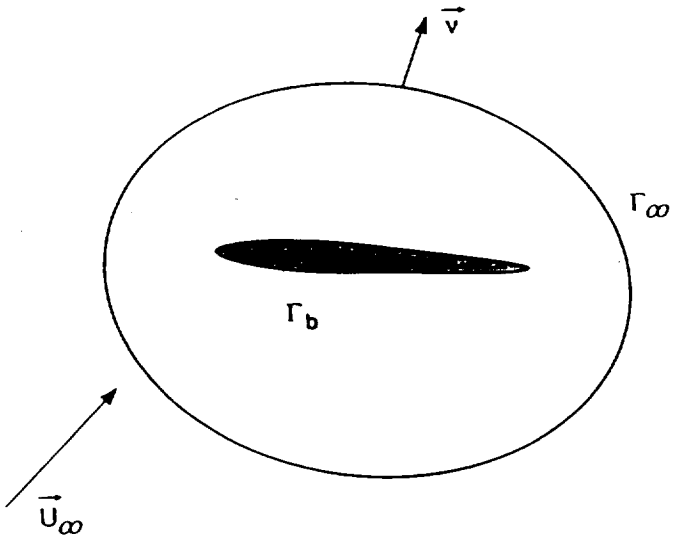


Fig. I.1. *The computational domain*

and μ_0 denote respectively, the characteristic density, velocity, length and diffusivity of the flow under consideration, and the Prandtl number $Pr = \dfrac{\mu_0 C_p}{k_0}$.

We consider the initial and boundary value problem (IBVP):

$$\begin{cases} \dfrac{\partial W}{\partial t} + \vec{\nabla} . \vec{\mathcal{F}}(W) = \dfrac{1}{Re} \vec{\nabla} . \vec{\mathcal{R}}(W) & \left(\vec{X}, t\right) \in \Omega \times \Re^+ \\ W\left(\vec{X}, 0\right) = W_0(\vec{X}) & \vec{X} \in \Omega \\ W\left(\vec{X}, t\right) = W_\Gamma(\vec{X}) & \vec{X} \in \Gamma = \partial\Omega \end{cases} \tag{7}$$

where W_0 and W_Γ are specified functions, and focus on finding a weak solution of (7) that is amenable to massively parallel computations.

I.2.2. Boundary conditions

We are mostly interested in external flows around airfoils. Therefore, we consider the case where the computational domain Ω is delimited by the boundary $\Gamma = \Gamma_b \cup \Gamma_\infty$. We denote by $\vec{v}$ the outward unit-normal at a given point of Γ (Fig. I.1).

In the far field, we assume that the viscous effects are negligible so that the flow is uniform. We adopt a formulation where the physical variables are non-dimensionalized. The free-stream vector W_∞ is given by:

$$\rho_\infty = 1 \quad \vec{U}_\infty = \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} \quad p_\infty = \frac{1}{\gamma M_\infty^2} \tag{8}$$

where α is the angle of attack and M_∞ is the free-stream Mach number. On the wall boundary Γ_b, we impose the no-slip condition and specify the temperature:

$$\vec{U} = \vec{0} \quad T = T_b \tag{9}$$

We do not impose any boundary condition on the density. Therefore, the total energy per unit of volume and the pressure on the wall are given by :

$$E = \rho C_v T_b \quad p = (\gamma - 1)E \tag{10}$$

I.3. Spatial discretization

I.3.1. Preliminary

The flow domain Ω is assumed to be a polygonal bounded region of $\Re^2$. Let $\mathcal{T}_h$ be a standard triangulation of Ω, and h the maximal length of the edges of $\mathcal{T}_h$. A vertex of a triangle Δ is denoted by S_i, and the set of its neighboring vertices by $K(i)$. At each vertex S_i, a cell C_i is constructed as the union of the subtriangles resulting from the subdivision by means of the medians of each triangle of $\mathcal{T}_h$ that is connected to S_i (Fig. A2). The boundary of C_i is denoted by ∂C_i, and the unit vector of the outward normal to ∂C_i by $\vec{\nu}_i = (\nu_{ix}, \nu_{iy})$. The union of all of the constructed cells forms a non-overlapping partition of the domain Ω:

$$\Omega = \bigcup_{i=1}^{ns} C_i \tag{11}$$

For each cell C_i, a characteristic function Ψ_i is defined as :

$$\Psi_i(\vec{X}) = \begin{cases} 1 & \text{if } \vec{X} \in C_i \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Also, the following discrete spaces are introduced:

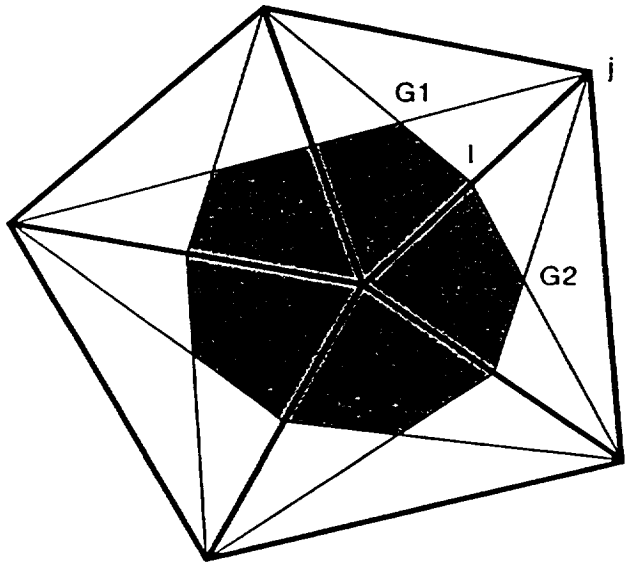


Fig. A2. *Cell definition in an unstructured grid*

$$\begin{aligned} \mathcal{V}_h &= \{v_h \mid v_h \in C^0(\Omega),\ v_h\,|_\Delta \in\ P_1,\ \forall \Delta \in \mathcal{T}_h\} \\ \mathcal{W}_h &= \{v_h \mid v_h \in L^2(\Omega),\ v_h\,|_{C_i} = v_i = \text{constant},\ i = 1, \ldots, ns\} \end{aligned} \tag{13}$$

where P_1 is the space of polynomials in two variables and of degree 1. Clearly, any function f belonging to $\mathcal{V}_h$ is uniquely determined by its values $f(S_i)$ at each vertex S_i, and can be expressed as:

$$f(\overrightarrow{X}) = \sum_{i=1,ns} f(S_i) N_i(\overrightarrow{X}) \tag{14}$$

where $\{N_i\}_{i=1}^{i=ns}$ is a basis of $\mathcal{V}_h$. Finally, it is noted that a natural bijection between the spaces $\mathcal{V}_h$ and $\mathcal{W}_h$ can be constructed as:

$$\forall f \in \mathcal{V}_h\ ,\ \ S(f(\overrightarrow{X})) = \sum_{i=1,ns} f(S_i) \Psi_i(\overrightarrow{X}) \tag{15}$$

3.2 Variational formulation and first order spatial approximations

A variational formulation of the IBVP (7) goes as follows:

$$\text{Find}\ \ W_h \in (\mathcal{V}_h)^4,\ \ \forall \varphi_h \in \mathcal{V}_h$$

$$\int_{\Omega} \frac{\partial W_h}{\partial t} \varphi_h dxdy + \int_{\Omega} \vec{\nabla} . \vec{\mathcal{F}}(W_h) \varphi_h dxdy = \frac{1}{Re} \int_{\Omega} \vec{\nabla} . \vec{\mathcal{R}}(W_h) \varphi_h dxdy \tag{16}$$

We construct a mixed finite volume/finite element (Galerkin) approximation for solving the above problem by introducing appropriate schemes for computing the left and right-hand-side integrals of (16). Chosing φ_h as the shape function N_i associated with the node S_i and applying the operator S to the left hand side of (16) leads to a mass-lumped variational approach which transforms the above equation into:

$$\int_{C_i} \frac{\partial W_h}{\partial t} dxdy + \int_{C_i} \vec{\nabla} . \vec{\mathcal{F}}(W_h) dxdy = \frac{1}{Re} \int_{Sup N_i} \vec{\nabla} . \vec{\mathcal{R}}(W_h) N_i dxdy \tag{17}$$

where $Sup N_i = \bigcup_{\Delta, S_i \in \Delta} \Delta$. Using Green's formula for the convective term and integrating by part the diffusive one leads to:

$$\int_{C_i} \frac{\partial W_h}{\partial t} dxdy + \int_{\partial C_i} \vec{\mathcal{F}}(W_h) . \vec{\nu}_i d\sigma = -\frac{1}{Re} \sum_{\Delta, S_i \in \Delta} \int_{\Delta} \vec{\mathcal{R}}(W_h) . \overrightarrow{\nabla N_i^{\Delta}} dxdy + \frac{1}{Re} \int_{\Gamma_h \cup \Gamma_\infty} \vec{\mathcal{R}}(W_h) . \vec{\nu}_i N_i d\sigma \tag{18}$$

where N_i^{Δ} is the restriction of N_i to triangle Δ. Finally, we drop the right hand side boundary integral as we enforce the viscous boundary conditions in a strong form on Γ_h and neglect the viscous effects on Γ_∞, so that equation (18) simplifies to:

$$\int_{C_i} \frac{\partial W_h}{\partial t} dxdy + \sum_{j \in K(i)} \int_{\partial C_{ij}} \vec{\mathcal{F}}(W_h).\vec{\nu}_{ij} d\sigma \; < 1 >$$
$$+ \int_{\partial C_i \cap \Gamma_h} \vec{\mathcal{F}}(\overline{W}_h).\vec{\nu}_i d\sigma \; < 2 >$$
$$+ \int_{\partial C_i \cap \Gamma_\infty} \vec{\mathcal{F}}(\overline{W}_h).\vec{\nu}_i d\sigma \; < 3 > \qquad (19)$$
$$= - - \frac{1}{Re} \sum_{\Delta, S_i \in \Delta} \int_{\Delta} \vec{\mathcal{R}}(W_h).\overrightarrow{\nabla N_i^{\Delta}} dxdy \; < 4 >$$

where $\overline{W}_h$ is the specified value of W_h at the boundaries.

The reader should note that the above formulation leads to a locally one-dimensional computation of each convective term, along the normal direction $\vec{\nu}$. For this purpose, the boundary ∂C_i of the cell C_i is split into bi-segments ∂C_{ij} which join the middle point of the edge $[S_i S_j]$ to the centroids of the triangles having both of S_i and S_j as vertices (Fig. A3), and the integral $< 1 >$ is evaluated as:

$$\sum_{j \in K(i)} \int_{\partial C_{ij}} \vec{\mathcal{F}}(W_h).\vec{\nu}_{ij} d\sigma = \sum_{j \in K(i)} \vec{\mathcal{F}}(\tilde{U}). \int_{\partial C_{ij}} \vec{\nu}_{ij} d\sigma \qquad (20)$$

where $\vec{\mathcal{F}}(\tilde{U})$ is some approximation of the convective flux computed at the interface between cells C_i and C_j.

Following Fezoui and Stoufflet [6], we choose $\vec{\mathcal{F}}(\tilde{U})$ to be a numerical flux function Φ associated with a first-order accurate upwind scheme (Van Leer [7]). It is denoted here by $H_{ij}^{(1)}$, where the superscript $^{(1)}$ emphasizes the first order accuracy, and can be written as:

$$H_{ij}^{(1)} = \Phi_{\mathcal{F}_{ij}}(W_i, W_j, \vec{\nu}_{ij}) \qquad (21)$$

where $W_i = W_h(S_i)$ and $W_j = W_h(S_j)$. For example, the following numerical flux functions can be used to construct $H_{ij}^{(1)}$:

- *Roe's Scheme* [8]

$$\Phi_{\mathcal{F}}^{R}\left(U, V, \vec{\nu}\right) = \frac{\mathcal{F}(U, \vec{\nu}) + \mathcal{F}(V, \vec{\nu})}{2} - d(U, V, \vec{\nu}) \qquad (22)$$

where $d\left(U, V, \vec{\nu}\right)$ is a numerical diffusivity defined as:

$$d\left(U, V, \vec{\nu}\right) = \mid \mathcal{A}\left(\tilde{W}, \vec{\nu}\right) \mid \frac{(V - U)}{2} \qquad (23)$$

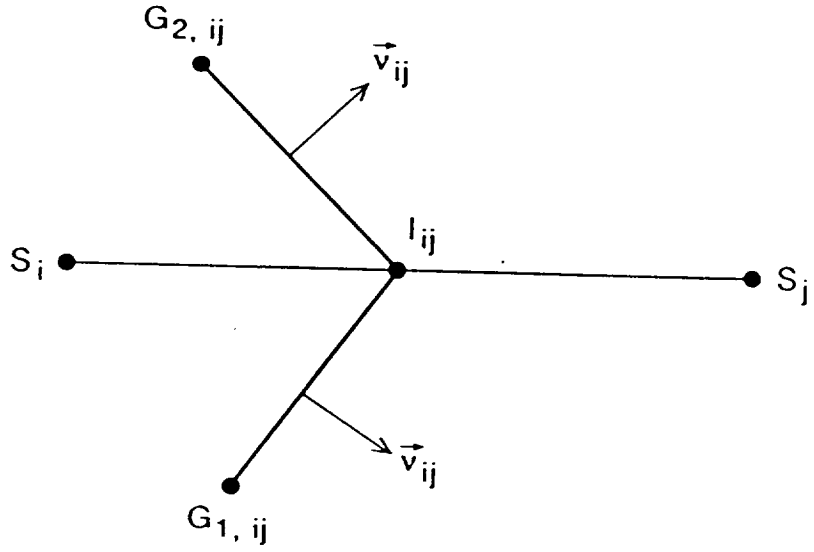


Fig. A3. *Splitting of* ∂C_{ij}

and $\tilde{W}$ is some mean value of U et V.

- *Steger and Warming's scheme* [9]

$$\Phi_{\mathcal{F}}^{SW}\left(U, V, \vec{\nu}\right) = \mathcal{A}^{+}\left(U, \vec{\nu}\right) U + \mathcal{A}^{-}\left(V, \vec{\nu}\right) V \tag{24}$$

where $\mathcal{A} = \mathcal{A}^{+} + \mathcal{A}^{-}$ and $\mid \mathcal{A} \mid = \mathcal{A}^{+} - \mathcal{A}^{-}$.

The viscous integral $< 4 >$ is evaluated via a classical Galerkin finite element $P1$ method which results in a centered scheme. Since the approximations of the physical variables are taken in $\mathcal{V}_h$, the components of the stress tensor and those of $\overrightarrow{\nabla N_i^{\Delta}}$ are constant in each triangle. The velocity vector in a triangle is computed as:

$$\vec{U}_{\Delta} = \frac{1}{3} \sum_{k=1, k \in \Delta}^{3} \vec{U}^{k} \tag{25}$$

Consequently, the viscous fluxes are evaluated as:

$$\sum_{\Delta, S_i \in \Delta} \int_{\Delta} \vec{\mathcal{R}}\left(W_h\right) . \overrightarrow{\nabla N_i^{\Delta}} dx dy = \sum_{\Delta, S_i \in \Delta} area(\Delta) \left(R_{\Delta} \frac{\partial N_i^{\Delta}}{\partial x} + S_{\Delta} \frac{\partial N_i^{\Delta}}{\partial y} \right) \tag{26}$$

where R_Δ and S_Δ are the constant values of $R(W)$ and $S(W)$ in the triangle Δ.

I.3.3. Higher order extension

The numerical integration with an upwind scheme described above leads to a spatial approximation that is only first-order accurate. Here, we focus on constructing a second-order accurate solution without changing the space of approximations. We develop a second-order scheme that is an extension of Van Leer's MUSCL method [7] to the case of unstructured meshes.

Usually, a second-order approximation requires the evaluation of the gradient of the solution at each vertex. Clearly, the gradient of a function v_h of $\mathcal{V}_h$ is constant in each element and discontinuous in the flow domain. Following the MUSCL method, one way to achieve second-order spatial accuracy is to evaluate the fluxes with extrapolated values W_{ij}, W_{ji} at the interface $\partial C_i \cap \partial C_j$. Basically, this leads to substituting $H_{ij}^{(1)}$ in the previous scheme by $H_{ij}^{(2)}$ which is given by:

$$\begin{aligned} H_{ij}^{(2)} &= \Phi_{\mathcal{F}_{ij}}(W_{ij}, W_{ji}, \overrightarrow{\nu}_{ij}) \\ W_{ij} &= W_i + \tfrac{1}{2}(\overrightarrow{\nabla} W)_i^\beta . \overrightarrow{S_i S_j} \\ W_{ji} &= W_j - \tfrac{1}{2}(\overrightarrow{\nabla} W)_j^\beta . \overrightarrow{S_i S_j} \end{aligned} \tag{27}$$

where the approximate nodal gradients $(\overrightarrow{\nabla} W)_{i,j}^\beta$ are obtained via a β-combination of centered and fully upwind gradients :

$$(\overrightarrow{\nabla W})_i^\beta = (1-\beta)(\overrightarrow{\nabla W})_i^{Cent} + \beta(\overrightarrow{\nabla W})_i^{Upw} \tag{28}$$

Here, a centered gradient $(\overrightarrow{\nabla W})_i^{Cent} = (\overrightarrow{\nabla W})^{\beta=0}$ can be chosen as any vector satisfying:

$$(\overrightarrow{\nabla W})_i^{Cent} . \overrightarrow{S_i S_j} = W_j - W_i \tag{29}$$

A nicely parallelizable scheme for computing the upwind gradients $(\overrightarrow{\nabla W})_i^{Upw}$ goes as follows. First, we note that $(\overrightarrow{\nabla W})_i^{Upw} = (\overrightarrow{\nabla W})^{\beta=1}$, and from (28) we derive:

$$(\overrightarrow{\nabla W})_i^{Upw} = 2(\overrightarrow{\nabla W})_i^{\beta=\frac{1}{2}} - (\overrightarrow{\nabla W})_i^{Cent} \tag{30}$$

We compute the half-upwind gradients ($\beta = \frac{1}{2}$) via a linear interpolation of the Galerkin gradients computed in each triangle of C_i, so that:

$$\left(\overrightarrow{\nabla W}\right)_i^{\beta=\frac{1}{2}} = \frac{\int_{C_i} \overrightarrow{\nabla W}|_\Delta dxdy}{\int_{C_i} dxdy} = \frac{1}{area(C_i)} \sum_{\Delta \epsilon C_i} \frac{area(T)}{3} \sum_{k=1,k\epsilon T}^{3} W^k \overrightarrow{\nabla \varphi_k} \tag{31}$$

Finally, we evaluate the nodal gradients using the following third-order biased scheme:

$$\begin{aligned}(\overrightarrow{\nabla W})_i^{\beta=\frac{1}{3}} &= \frac{2}{3}(\overrightarrow{\nabla W})_i^{\beta=0} + \frac{1}{3}(\overrightarrow{\nabla W})_i^{\beta=1} \\ &= \frac{2}{3}(\overrightarrow{\nabla W})_i^{\beta=0} + \frac{1}{3}\left(2(\overrightarrow{\nabla W})_i^{\beta=\frac{1}{2}} - (\overrightarrow{\nabla W})_i^{\beta=0}\right) \\ &= \frac{1}{3}(\overrightarrow{\nabla W})_i^{\beta=0} + \frac{2}{3}(\overrightarrow{\nabla W})_i^{\beta=\frac{1}{2}}\end{aligned} \tag{32}$$

I.3.4. Boundary conditions

The second term $< 2 >$ and the third term $< 3 >$ of the right-hand side of (19) contain the physical boundary conditions. These are represented by the vector $\overline{W}_h$ which involves quantities that depend on the interior values of W_h, and quantities that are determined by the physical boundary conditions.

Wall boundary : the no-slip condition is enforced in a strong form (9, 10) so that the corresponding boundary integral $< 2 >$ does not need to be evaluated.

Inflow and outflow boundaries : at these boundaries, a precise set of compatible exterior data which depend on the flow regime and the velocity direction must be specified. For that purpose, a *plus-minus* flux splitting is applied between exterior data and interior values. More precisely, the boundary integral $< 3 >$ is evaluated using a non-reflective version of the flux-splitting of Steger and Warming [9] :

$$\int_{\partial C_i \cap \Gamma_\infty} \mathcal{F}(\overline{W}_h).\overrightarrow{\nu}_i d\sigma = \mathcal{A}^+(W_i, \overrightarrow{\nu}_{i\infty}).W_i + \mathcal{A}^-(W_i, \overrightarrow{\nu}_{i\infty}).W_\infty \tag{33}$$

I.4. Time discretization

The resulting semi-discrete fluid flow equations can be written as:

$$\frac{dW}{dt} + \psi(W) = 0 \tag{34}$$

Because it lends itself to massive parallelism, the explicit Runge-Kutta method is selected for integrating the above equations. A 3-step variant is used here. It is summarized as :

$$\begin{cases} W^{(0)} = W^n \\ W^{(k)} = W^{(0)} - \dfrac{\Delta t}{4-k}\psi(W^{(k-1)}) \qquad k = 1, 2, 3 \\ W^{n+1} = W^{(3)} \end{cases} \tag{35}$$

The above scheme is often referred to as the low-storage Runge-Kutta method as only the solution at substep $\alpha - 1$ is used to compute the one at substep α. It is third-order accurate in the linear case, but only second-order accurate in our case.

I.5. Parallel implementation on the Connection Machine CM-2

Clearly, expressions (19) and (27-35) reveal that both the spatial and temporal integrations are in principle nicely parallelizable. In this section, our interest lies in investigating the most efficient way to implement these computations on a Single Instruction Multiple Data (SIMD) massively parallel computer such as the Connection Machine CM-2. Special care is given to interprocessor communication because mesh irregularities: (a) inhibit the exploitation of the NEWS grid, so that the relatively slow router must be used, and (b) induce a different amount of communication steps within each processor, which is not particularly desirable on a SIMD machine. Rather than overviewing the CM-2, we refer the reader to the technical summary of Thinking Machines [10] for architectural details, and to Farhat, Sobh, and Park [3] for an in-depth analysis of interprocessor communication on the CM-2 when computing over an irregular mesh.

I.5.1. Parallel data structure

Behind the performance of any parallel algorithm lies the choice of the corresponding parallel data structure. The latter is closely related to both the entity and the task to be assigned to each processor. Therefore, all of the computational, communication and memory requirements should be considered before the distributed data structure is determined. For the mixed finite volume/finite element method presented here, we consider four candidates for a fundamental entity (Fig. A4):

- the vertex S_i,
- the edge E_{ij} joining the vertices S_i and S_j,
- the element (here the triangle) Δ_{ijk} connecting the vertices S_i, S_j and S_k,
- and the cell C_i defined in Section 3.1.

Memory considerations

While regular grids are most often characterized (in terms of memory requirements) by their number of vertices N_V, irregular triangular grids can be also characterized by either their number of elements N_Δ, or by their number of edges N_E. Here, we assume for simplicity that $\mathcal{T}_h$ is characterized by its number of vertices. Euler's relations for a triangulation state that :

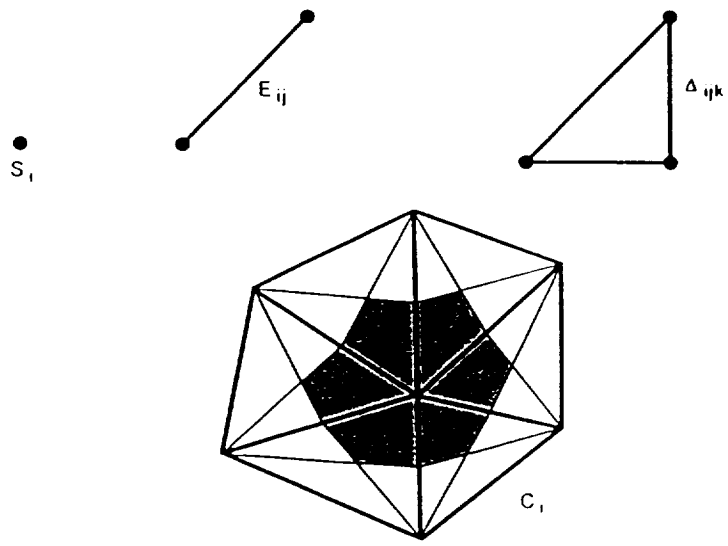


Fig. A4. *Fundamental entity candidates*

$$\begin{aligned} N_V + N_\Delta - N_E &= 1 \\ 2N_E - N_{BV} &= 3N_\Delta \end{aligned} \tag{36}$$

where N_{BV} denotes the number of vertices at the boundary of the triangulation. This implies that :

$$N_\Delta \approx 2N_V \quad \text{and} \quad N_E \approx 3N_V \tag{37}$$

Therefore, if $\mathcal{T}_h$ is designed, for example, so that its number of vertices matches a given Connection Machine size, the VP ratio associated with each data structure candidate varies as indicated below:

	Vertex	**Edge**	**Element**	**Cell**
VPR	1	3	2	1

The reader should note that for the edge case, the machine automatically selects a VP ratio of 4, since it is the closest power of two to the theoretical VPR. Clearly, the vertex and cell entities are the best candidates on the sole basis of efficient memory usage.

Operation count

The numerical algorithms discussed in Section 2 and Section 3 can be organized around three basic computational steps :

(Step a) evaluation of the Galerkin gradients (32),

(Step b) evaluation of the diffusive fluxes (26),

(Step c) and evaluation of the convective fluxes (27).

While **Step (c)** is most efficiently performed using edge-wise computations, **Step (a)** and **Step (b)** are inherently element-level calculations. Therefore, whatever fundamental entity is selected, it must contain both edge and element information, which rules out the edge E_{ij} data structure.

On the other hand in an element-based partition, every triangle Δ_{ijk} provides direct access to all of the three edges E_{ij}, E_{jk} and E_{ki}. However in that case, two VP sets must be used; one containing N_Δ processors which store triangle related data (geometrical data), and another one containing N_V processors which store vertex related data (physical data). Otherwise, if only one set of virtual processors is used and assigned to both triangle and vertex data, a nodal result would be duplicated in as many processors as there are triangles connected to that vertex.

The vertex entity S_i is an effective candidate only when augmented with the auxiliary data structures that can handle the data associated with the elements and edges connected to a given vertex — that is, when transformed into a cell data structure.

Finally, we note that the cell entity stores both vertex and element data, and therefore provides access to all of vertex, element and edge information. Consequently, only element and cell partitions are retained for further discussions.

Next, we evaluate the operation count for each of **Step (a)**, **Step (b)** and **Step (c)**, assuming an element- or cell-based data structure. We denote by C_c^E and C_{ab}^Δ, the number of arithmetic operations associated with one edge computation during **Step (c)**, and with one triangle computation during **Step (a)** and **Step (b)**, respectively. The computational complexities characterizing the two retained candidates are tabulated below.

	Element	**Cell**
Step (c)	$2 \times C_c^E$	$2 \times C_c^E$
Step (a) + Step (b)	C_{ab}^Δ	$3 \times C_{ab}^\Delta$

In both an element- and cell-based partition, an edge is shared by two virtual processors, so that the flux $H_{ij}^{(2)}$ across $[S_i S_j]$ is computed twice. Only an edge partition would eliminate these redundant computations, but that choice has already been eliminated. In a cell-based partition, a triangle Δ_{ijk} is shared by three virtual processors, and therefore additional redundant computations are generated.

Communication costs

The computational steps discussed above require four communication steps denoted here by **(c1)**, **(c2)**, **(c3)**, and **(c4)**. These are discussed below for the element and cell parallel data structures.

First, we consider the case of an element-based partition. During the first communication step **(c1)**, each virtual processor assigned to a triangle Δ_{ijk} gets the physical states at vertices S_i, S_j

and S_k from neighboring processors. Then, the computations in **Step (a)** and **Step (b)** are carried out. During the second communication step **(c2)**, the element-wise results are sent back to the virtual processors holding vertex data. The latter virtual processors use these values to compute the nodal gradients (32) and diffusive fluxes (26). In step **(c3)** the nodal gradients are communicated to neighboring processors. Next, each virtual processor evaluates three second-order convective fluxes (15) across the three edges connected by triangle Δ_{ijk}. During the last communication step **(c4)**, the edge-wise fluxes are sent to the virtual processors holding vertex data.

Communication with a cell-based partition is more complex, as each cell may have a different number of neighbors. However, fewer communication steps are needed because each virtual processor stores within its local memory all of the element-wise values that are necessary for the evaluation of the nodal gradients and the diffusive fluxes, as well as the elemental convective fluxes.

The communication count associated with the four steps **(c1)** to **(c4)** is tabulated below for each of the two retained data structure candidates. N_{neigh}^{max} denotes the maximum number of neighboring cells.

	Element	**Cell**
(c1)	3	N_{neigh}^{max}
(c2)	3	0
(c3)	3	N_{neigh}^{max}
(c4)	6	0

Selected candidate

The operation and communication counts are summarized below for both the element and cell data structures. Equations (36) are used to express the results in terms of the number of vertices in the mesh.

	Element	**Cell**
Operation count	$(6 \times C_c^E + 2 \times C_{ab}^{\Delta}) \times N_V$	$(6 \times C_c^E + 6 \times C_{ab}^{\Delta}) \times N_V$
Communication count	$30 \times N_V$	$12 \times N_V$

Clearly, redundant arithmetic operations can be avoided only at the expense of additional communication characterized by an irregular pattern, which is usually not beneficial on a massively parallel processor such as the CM-2. Therefore, we have chosen the cell-based parallel data structure and have accepted the additional cost of redundant flux computations. Hammond and Barth [5] have invoked a graph theory result due to Chrobak and Eppstein [17] to eliminate redundant edge-based flux computations for Euler flows. This result states that for any planar graph, there exists an orientation of the edges such that no vertex has more than three edges directed out from it. This means that there exists a cell partition where no processor needs to compute the convective fluxes across more than three edges of the computational cell. However, this graph theory result does not apply for our viscous computations because these also include element-based operations.

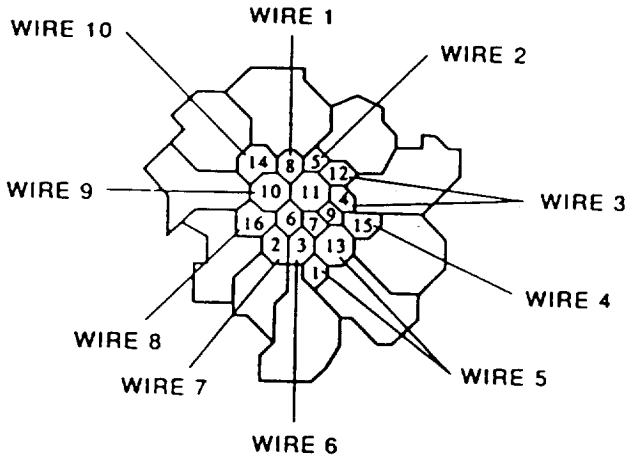


Fig. A5. *Grid decomposition with reduced wire-contention*

I.5.2. Grid decomposition and processor mapping

Efficiency in arbitrary communication on the CM-2 requires the minimization of both the "hammering" on the router — that is, wire contention, and the distance that information has to travel — that is, the number of hops between the sender and receiver processors. Here, this implies that : (a) adjacent cells must be assigned, as much as possible, to directly connected processors or processors that are lying in directly connected chips, and (b) contention for the wire connecting neighboring chips must be reduced.

In a first step, the unstructured grid is decomposed into a series of subgrids each containing 16 adjacent numerical cells. Each subgrid is assigned to a certain CM-2 chip that is subsequently identified, so that adjacent cells within a subgrid are assigned to directly connected processors lying in the same chip. As a result, off-chip communication is needed only across the subgrid boundaries. Wire contention is reduced if each of the defined subgrids is surrounded by the largest possible number of neighboring subgrids. Indeed, wherever a subgrid boundary is shared with several other subgrids, off-chip communication is split between distinct chips and is distributed across several of the available inter-chip wires (Fig. A5). On the other hand, if for example a subgrid is adjacent only to two other subgrids, a maximum of two wires can be used during off-chip communication, which may create a severe wire contention that would serialize communication and significantly increase its cost. Here, we use the mesh decomposer of Farhat [11] which has proven to be very effective at reducing wire contention on the CM-2 (Farhat, Sobh and Park [3]).

The next step is to reduce the distance that information has to travel during off-chip communication,

that is when data is exchanged between centers of cells that are assigned to processors lying on different chips. This can be achieved by assigning adjacent subgrids as far as possible to directly connected chips. A combinatorial optimization-like procedure known as *Simulated Annealing* (see, for example, Flower, Otto and Salama [12]) is probably the most popular technique for tackling this mapping problem. However, it is a very expensive procedure which has often proved to be impractical. Alternative heuristic-based schemes have been developed by several authors including Bokhari [13], Farhat [14], and recently Hammond and Schreiber [15]. In this work, we have adopted the mapper of reference [14]. It is based on a combined greedy/divide and conquer approach and is tuned for hypercube topologies.

A detailed analysis of interprocessor communication on the CM-2 for unstructured grids can be found in Farhat, Sobh and Park [3]. In that reference, it is shown that mesh irregularities induce an MIMD (Multiple Instruction Multiple Data) style of programming for the communication phase which dominates the cost of communication. It is also suggested that since the irregular pattern of communication is fixed in time, a considerable improvement can be achieved if that pattern is evaluated during the first time step, then compiled or stored in the CM-2 for re-use in subsequent time steps. However, no software was available at that time for validating the proposed communication strategy. Recently, a communication compiler prototype has become available (Dahl [16]) and can be used for storing the routing pattern. In Section 6, we report on its performance.

I.6. Numerical Experiments

(This Section reports on numerical experiments on the CM-2 and Cray 2. Since airfoil problems are of limited important for the present research, they are not presented here.)

I.7. Closure

Mixed finite volume/finite element spatial schemes for fully unstructured grids are developed and implemented on the CM-2, and applied to the simulation of two-dimensional viscous flows. Second-order accuracy in the discretization of the convective fluxes is achieved through a Monotonic Upwind Scheme for Conservation Laws (MUSCL) technique. The diffusive fluxes are computed using a classical Galerkin finite element method, and the resulting semi-discrete equations are time integrated with an explicit Runge-Kutta algorithm.

A strategy for mapping thousands of processors onto an unstructured grid is presented. Its key elements are given by the selection of an appropriate parallel data structure, the careful partitioning of a given unstructured grid into specific subgrids, and the mapping of each individual processor onto an entity of these subgrids. Whenever the communication patterns are compiled during the first time step, the total time elapsed in interprocessor communication using the router is drastically reduced to represent only 15% of the total CPU time of the simulation.

References

[1] A. Saati, S. Biringen and C. Farhat, "Solving Navier-Stokes Equations on a Massively Parallel Processor: Beyond the One Gigaflop Performance," *Int. J. Supercomp. Appl.*, Vol. 4, No. 1, pp. 72-80, (1990).

[2] S. Lanteri, C. Farhat and L. Fezoui, "Structured Compressible Flow Computations on the Connection Machine," *INRIA Report No. 1322*, (1990).

[3] C. Farhat, N. Sobh and K. C. Park, "Transient Finite Element Computations on 65536 Processors : The Connection Machine," *Int. J. Num. Meth. Eng.*, Vol. 30, pp. 27-55, (1990).

[4] R. A. Shapiro, "Implementation of an Euler/Navier-Stokes Finite Element Algorithm on the Connection Machine," *AIAA Paper 91-0483*, 29th Aerospace Sciences Meeting, Reno (1991).

[5] S. Hammond and T. Barth, "An Efficient Massively Parallel Euler Solver for Unstructured Grids," *AIAA Paper 91-0441*, 29th Aerospace Sciences Meeting, Reno, Nevada (1991).

[6] L. Fezoui and B. Stoufflet, "A Class of Implicit Upwind Schemes for Euler Simulations with Unstructured Meshes," *J. Comp. Phys.*, Vol. 84, pp. 174-206, (1989).

[7] B. Van Leer, "Towards the Ultimate Conservative Difference Scheme V: a Second-Order Sequel to Goudonov's Method," *J. Comp. Phys.*, Vol. 32, (1979).

[8] P. L. Roe, "Approximate Riemann Solvers, Parameters Vectors and Difference Schemes," *J. Comp. Phys.*, Vol. 43, pp. 357-371, (1981).

[9] J. Steger and R. F. Warming, "Flux Vector Splitting for the Inviscid Gas Dynamic with Applications to Finite-Difference Methods," *J. Comp. Phys.*, Vol. 40, No. 2, pp. 263-293, (1981).

[10] Thinking Machines Corporation, "Connection Machine Model CM-2: Technical Summary," Version 6.0, (1990).

[11] C. Farhat, "A Simple and Efficient Automatic Finite Element Mesh Domain Decomposer," *Comp. & Struct.*, Vol. 28, No. 5, pp. 579-602, (1988).

[12] J. W. Flower, S. W. Otto and M. C. Salama, "A Preprocessor for Irregular Finite Element Problems," *CalTech/JPL Report C3P-292*, (1986).

[13] S. H. Bokhari, "On the Mapping Problem," *IEEE Trans. Comp.*, Vol. C-30, No. 3, pp. 207-214, (1981).

[14] C. Farhat, "On the Mapping of Massively Parallel Processors Onto Finite Element Graphs," *Comp. & Struct.*, Vol. 32, No. 2, pp. 347-354, (1989).

[15] S. Hammond and R. Schreiber, "Mapping Unstructured Grid Problems to the Connection Machine," *RIACS Technical Report 90.22*, (1990).

[16] E. D. Dahl, "Mapping and Compiling Communication on the Connection Machine System," *Proc. Distr. Mem. Comp. Conf.*, Charleston, (1990).

[17] M. Chrobak and D. Epstein, "Planar Orientations with Low Out-Degree and Compaction of Adjacency Matrices," *Theor. Comp. Sci., To appear*, (1990).

Appendix II

Parallel Staggered Algorithms for the Solution of Three-Dimensional Aeroelastic Problems

Summary

This Appendix outlines recent developments in the solution of large-scale three-dimensional (3D) nonlinear aeroelastic problems on high performance, massively-parallel computational platforms. Developments include a three-field arbitrary Lagrangian-Eulerian (ALE) finite volume/element formulation for the coupled fluid/structure problem, a geometric conservation law for 3D flow problems with moving boundaries and unstructured deformable meshes, and the solution of the corresponding coupled semi-discrete equations with partitioned heterogeneous procedures. We present a family of mixed explicit/implicit staggered solution algorithms, and discuss them with particular reference to accuracy, stability, subcycling, and parallel processing. We describe a general framework for the solution of coupled aeroelastic problems on heterogeneous and/or parallel computational platforms, and illustrate it with some preliminary numerical investigations of transonic aerodynamics and aeroelastic responses on several massively parallel computers, including the iPSC-860, Paragon XP/S, Cray T3D, and IBM SP2. The work described here was carried out by P.-S. Chen, M. Lesoinne and P. Stern under supervision from Professor C. Farhat.

II.1. INTRODUCTION

In order to predict the aeroelastic behavior of flexible structures in fluid flows, the equations of motion of the structure and the fluid must be solved simultaneously. Because the position of the structure determines at least partially the boundaries of the fluid domain, it becomes necessary to perform the integration of the fluid equations on a moving mesh. Several methods have been proposed for this purpose. Among them we note the Arbitrary Lagrangian Eulerian (ALE) formulation [7], dynamic meshes [3], the co-rotational approach [8,11,24], and the Space-Time finite element method [38].

Although the aeroelastic problem is usually viewed as a two-field coupled problem (see for example, Guruswamy [22]), the moving mesh can be viewed as a pseudo-structural system with its own dynamics, and therefore, the coupled aeroelastic system can be formulated as a three-field problem, the components of which are the fluid, the structure, and the dynamic mesh [26]. The semi-discrete equations that govern this three-way coupled problem can be written as follows.

$$
\begin{aligned}
\frac{\partial}{\partial t}(\mathbf{V}(\mathbf{x}, t)\, \mathbf{w}(t)) + \mathbf{F}^c(\mathbf{w}(t), \mathbf{x}, \dot{\mathbf{x}}) &= \mathbf{R}(\mathbf{w}(t)) \\
\mathbf{M}\frac{\partial^2 \mathbf{q}}{\partial t^2} + \mathbf{f}^{int}(\mathbf{q}) &= \mathbf{f}^{ext}(\mathbf{w}(t), \mathbf{x}) \\
\widetilde{\mathbf{M}}\frac{\partial^2 \mathbf{x}}{\partial t^2} + \widetilde{\mathbf{D}}\frac{\partial \mathbf{x}}{\partial t} + \widetilde{\mathbf{K}}\mathbf{x} &= \mathbf{K}_c\, \mathbf{q}
\end{aligned}
\tag{1}
$$

where t denotes time, $\mathbf{x}$ is the position of a moving fluid grid point, $\mathbf{w}$ is the fluid state vector, $\mathbf{V}$ results from the flux-split finite-element (FE) and finite-volume (FV) discretization of the fluid equations, $\mathbf{F}^c$ is the vector of convective ALE fluxes, $\mathbf{R}$ is the vector of diffusives fluxes, $\mathbf{q}$ is the structural displacement vector, $\mathbf{f}^{int}$ denotes the vector of internal forces in the structure, $\mathbf{f}^{ext}$ the vector of external forces, $\mathbf{M}$ is the FE mass matrix of the structure, $\widetilde{\mathbf{M}}$, $\widetilde{\mathbf{D}}$ and $\widetilde{\mathbf{K}}$ are fictitious mass, damping and stiffness matrices associated with the moving fluid grid, and $\mathbf{K}_c$ is a transfer matrix that describes the action of the motion of the structural side of the fluid/structure interface on the fluid dynamic mesh.

For example, $\widetilde{\mathbf{M}} = \widetilde{\mathbf{D}} = 0$, and $\widetilde{\mathbf{K}} = \widetilde{\mathbf{K}}^R$ where $\widetilde{\mathbf{K}}^R$ is a rotation matrix corresponds to a rigid mesh motion of the fluid grid around an oscillating airfoil, and $\widetilde{\mathbf{M}} = \widetilde{\mathbf{D}} = 0$ includes as particular cases the spring-based mesh motion scheme introduced by Batina [3], and the continuum based updating strategy described by Tezduyar [38]. In general, $\mathbf{K}^c$ and $\tilde{\mathbf{K}}$ are designed to enforce continuity between the motion of the fluid mesh and the structural displacement and/or velocity at the fluid/structure boundary $\Gamma_{F/S}(t)$:

$$
\begin{aligned}
x(t) &= q(t) \quad \text{on } \Gamma_{F/S}(t) \\
\dot{x}(t) &= \dot{q}(t) \quad \text{on } \Gamma_{F/S}(t)
\end{aligned}
\tag{2}
$$

Each of the three components of the coupled problem described by Eqs. (1) has different mathematical and numerical properties, and distinct software implementation requirements. For Euler and Navier-Stokes flows, the fluid equations are nonlinear. The structural equations and the semi-discrete equations governing the pseudo-structural fluid grid system may be linear or nonlinear. The matrices resulting from a linearization procedure are in general symmetric for the structural problem, but they are typically unsymmetric for the fluid problem. Moreover, the nature of the coupling in Eqs. (1) is implicit rather than explicit, even when the fluid mesh motion is ignored. The fluid and the structure interact only at their interface, via the pressure and the motion of the physical interface. However, for Euler and Navier-Stokes compressible flows, the pressure variable cannot be easily isolated neither from the fluid equations nor from the fluid state vector $\mathbf{w}$. Consequently, the numerical solution of Eqs. (1) via a fully coupled monolithic scheme is not only computationally challenging, but unwieldy from the standpoint of software development management.

Alternatively, Eqs. (1) can be solved via partitioned procedures [4,9,32], the simplest realization of which are the staggered procedures [31]. This approach offers several appealing features, including the ability to use well established discretization and solution methods within each discipline,

simplification of software development efforts, and preservation of software modularity. Traditionally, transient aeroelastic problems have been solved via the simplest possible staggered procedure whose typical cycle can be described as follows: a) advance the structural system under a given pressure load, b) update the fluid mesh accordingly, and c) advance the fluid system and compute a new pressure load [5,6,35,36]. Some investigators have advocated the introduction of a few predictor/corrector iterations within each cycle of this three-step staggered integrator in order to improve accuracy [37], especially when the fluid equations are nonlinear and treated implicitly [34]. Here we focus on the design of a broader family of partitioned procedures where the fluid flow is integrated using an explicit scheme, and the structural response is advanced using an implicit one. We address issues pertaining to numerical stability, subcycling, accuracy v.s. speed trade-offs, implementation on heterogeneous computing platforms, and inter-field as well as intra-field parallel processing.

We begin in Section II.2 with the discussion of a geometric conservation law (GCL) for the finite-volume approximation of three-dimensional flows with moving boundaries. In Section II.3 we introduce a partitioned solution procedure where the fluid flow is time-integrated using an explicit scheme while the structural response is advanced using an implicit scheme. This particular choice of mixed time-integration is motivated by the following facts: (a) the aeroelastic response of a structure is often dominated by low frequency dynamics, and therefore is most efficiently predicted by an implicit time-integration scheme, and (b) we have previously developed a massively parallel explicit FE/FV Navier-Stokes solver that we wish to re-use for aeroelastic computations. Two-dimensional versions of this solver have been described by Farhat and coworkers [10,12,25].

In practice, the stability limit of this partitioned procedure has proved to be governed only by the critical time-step of the explicit fluid solver. In Section II.4, we describe a subcycling procedure that does not limit the stability properties of a partitioned time-integrator. In Section II.5, we address important issues related to inter-field parallelism and design variants of the algorithm presented in Section II.3 that allow advancing simultaneously the fluid and structural systems. Section II.6 focuses on the implementation of staggered procedures on distributed and/or heterogeneous computational platforms. Finally, Section II.7 illustrates the work presented herein with some preliminary numerical results on four parallel computers: Intel iPSC-860, Intel Paragon XP/S, Cray T3D, and IBM SP2. These results pertain to the response of an axisymmetric engine model and of a 3D wing in a transonic airstream.

II.2. A GLOBAL CONSERVATION LAW FOR ALE-BASED FV METHODS

II.2.1. Semi-discrete flow equations

Let $\Omega(t) \subset \mathcal{R}^3$ be the flow domain of interest, and $\Gamma(t)$ be its moving and deforming boundary. For simplicity, we map the instantaneous deformed domain on the reference domain $\Omega(0)$ as follows:

$$x = x(\xi, t), \qquad t = \tau \tag{3}$$

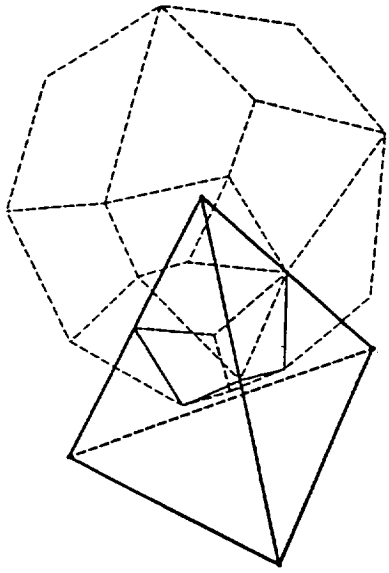


Figure II.1. A three-dimensional unstructured FV cell

The ALE conservative form of the equations describing Euler flows can be written as:

$$\left.\frac{\partial(JW)}{\partial t}\right|_{\xi} + J\nabla_x.\mathcal{F}^c(W) = 0, \qquad \mathcal{F}^c(W) = \mathcal{F}(W) - \dot{x}W \tag{4}$$

where $J = \det(dx/d\xi)$ is the Jacobian of the frame transformation $\xi \rightarrow x$, W is the fluid state vector, $\mathcal{F}^c$ denotes the convective ALE fluxes, and $\dot{x} = \frac{\partial x}{\partial \tau}|_{\xi}$ is the ALE grid velocity, which may be different from the fluid velocity and from zero. The fluid volume method for unstructured meshes relies on the discretization of the computational domain into control volumes or cells C_i, as illustrated in Figure II.1.

Because in an ALE formulation the cells move and deform in time, the integration of Eq. (4) is first performed over the reference cell in the ξ space

$$\int_{C_i(0)} \left.\frac{\partial(JW)}{\partial t}\right|_{\xi} d\Omega_{\xi} + \int_{C_i(0)} J\nabla_x.\mathcal{F}^c(W)\, d\Omega_{\xi} = 0 \tag{5}$$

Note that in Eq. (5) above the time derivative is evaluated at a constant ξ; hence it can be moved outside of the integral sign to obtain

$$\frac{d}{dt}\int_{C_i(0)} W\, J\, d\Omega_{\xi} + \int_{C_i(0)} \nabla_x.\mathcal{F}^c(W)\, J\, d\Omega_{\xi} = 0 \tag{6}$$

Switching back to the time varying cells, we have

$$\frac{d}{dt}\int_{C_i(t)} W\, d\Omega_x + \int_{C_i(t)} \nabla_x.\mathcal{F}^c(W)\, d\Omega_x = 0 \tag{7}$$

Finally, integrating by parts the last term yields the integral equation

$$\frac{d}{dt}\int_{C_i(t)} W\, d\Omega_x + \int_{\partial C_i(t)} \mathcal{F}^c(W).\vec{n}\, d\sigma = 0 \tag{8}$$

A key component of a FV method is the following approximation of the flux through the cell boundary $\partial C_i(t)$:

$$F_i(\mathbf{w}, \mathbf{x}, \dot{\mathbf{x}}) = \sum_j \int_{\partial C_{i,j}(\mathbf{x})} (\mathcal{F}^c_+(W_i, \dot{x}) + \mathcal{F}^c_-(W_j, \dot{x}))\, \vec{n}\, d\sigma \tag{9}$$

where W_i denotes the average value of W over the cell C_i, $\mathbf{w}$ is the vector formed by the collection of W_i, and $\mathbf{x}$ is the vector of time dependent nodal positions. The numerical flux functions $\mathcal{F}^c_+$ and $\mathcal{F}^c_-$ are designed to make the resulting system stable. An example of such functions may be found in Ref. [1]. For consistency, these numerical fluxes must verify

$$\mathcal{F}^c_+(W, \dot{x}) + \mathcal{F}^c_-(W, \dot{x}) = \mathcal{F}^c(W) \tag{10}$$

Thus, the governing discrete equation is:

$$\frac{d}{dt}(V_i W_i) + F_i(\mathbf{w}, \mathbf{x}, \dot{\mathbf{x}}) = 0 \tag{11}$$

where

$$V_i = \int_{C_i(t)} d\Omega_x \tag{12}$$

is the volume of cell C_i. Collecting all Eqs. (11) into a single system yields:

$$\frac{d}{dt}(\mathbf{V}\mathbf{w}) + \mathbf{F}(\mathbf{w}, \mathbf{x}, \dot{\mathbf{x}}) = 0 \tag{13}$$

where $\mathbf{V}$ is the diagonal matrix of the cell volumes, $\mathbf{w}$ is the vector containing all state variables W_i, and $\mathbf{F}$ is the collection of the ALE fluxes F_i.

II.2.2. A Geometric Conservation Law

Let Δt and $t^n = n\Delta t$ denote the chosen time-step and the n-th time-station, respectively. Integrating Eq. (11) between t^n and t^{n+1} leads to

$$\begin{aligned}\int_{t^n}^{t^{n+1}} \frac{d}{dt}(V_i(\mathbf{x}) W_i) dt + \int_{t^n}^{t^{n+1}} F_i(\mathbf{w}, \mathbf{x}, \dot{\mathbf{x}}) \\ = V_i(\mathbf{x}^{n+1}) W_i^{n+1} - V_i(\mathbf{x}^n) W_i^n \\ + \int_{t^n}^{t^{n+1}} F_i(\mathbf{w}, \mathbf{x}, \dot{\mathbf{x}}) = 0\end{aligned} \tag{14}$$

The most important issue in the solution of the first of Eqs. (1) via an ALE method is the proper evaluation of $\int_{t^n}^{t^{n+1}} F_i(\mathbf{w}, \mathbf{x}, \dot{\mathbf{x}})$ in Eq. (14). In particular, it is crucial to establish where the fluxes must be integrated: on the mesh configuration at $t = t^n(\mathbf{x}^n)$, on the configuration at $t = t^{n+1}(\mathbf{x}^{n+1})$, or in between these two configurations. The same questions also arise as to the choice of the mesh velocity vector $\dot{\mathbf{x}}$.

Let W^* denote a given uniform state of the flow. Clearly, a proposed solution method cannot be acceptable unless it conserves the state of a uniform flow. Substituting $W_k^n = W_k^{n+1} = W^*$ in Eq. (14) gives:

$$(V_i^{n+1} - V_i^n)W^* + \int_{t^n}^{t^{n+1}} F_i(\mathbf{w}^*, \mathbf{x}, \dot{\mathbf{x}})\, dt = 0 \tag{15}$$

in which $\mathbf{w}^*$ is the vector of the state variables when $W_k = W^*$ for all k. From Eq. (9) it follows that:

$$F_i(\mathbf{w}^*, \mathbf{x}, \dot{\mathbf{x}}) = \int_{\partial C_i(\mathbf{X})} (\mathcal{F}(W^*) - \dot{x} W^*)\, d\sigma \tag{16}$$

Given that the integral on a closed boundary of the flux of a constant function is identically zero we must have

$$\int_{\partial C_i(\mathbf{X})} \mathcal{F}(W^*)\, d\sigma = 0 \tag{17}$$

it follows that

$$F_i(\mathbf{w}^*, \mathbf{x}, \dot{\mathbf{x}}) = -\int_{\partial C_i(\mathbf{X})} \dot{x} W^*\, d\sigma \tag{18}$$

Hence, substituting Eq. (18) into Eq. (15) yields

$$(V_i(\mathbf{x}^{n+1}) - V_i(\mathbf{x}^n))W^* - (\int_{t^n}^{t^{n+1}} \int_{\partial C_i(\mathbf{X})} \dot{x}\, d\sigma\, dt)W^* = 0 \tag{19}$$

which can be rewritten as

$$\boxed{(V_i(\mathbf{x}^{n+1}) - V_i(\mathbf{x}^n)) = \int_{t^n}^{t^{n+1}} \int_{\partial C_i(\mathbf{X})} \dot{x}\, d\sigma\, dt} \tag{20}$$

Eq. (20) must be verified by any proposed ALE mesh updating scheme. We refer to this equation as the geometric conservation law (GCL) because: (a) it can be identified as integrating exactly the volume swept by the boundary of a cell in a FV formulation, (b) its principle is similar to the GCL condition that was first pointed out by Thomas and Lombard [39] for structured grids treated by finite difference schemes. More specifically, this law states that the change in volume of each cell between t^n and t^{n+1} must be equal to the volume swept by the cell boundary during the time-step $\Delta t = t^{n+1} - t^n$. Therefore, the updating of $\mathbf{x}$ and $\dot{\mathbf{x}}$ cannot be based on mesh distorsion issues alone when using ALE solution schemes.

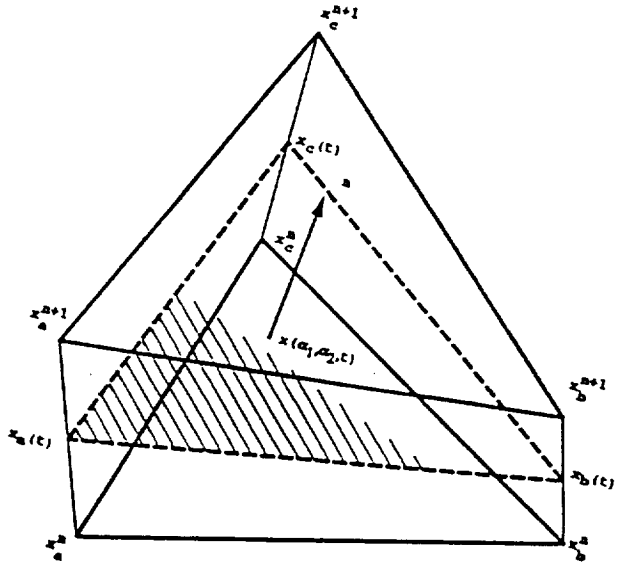


Figure II.2. Parametrization of a moving triangular facet.

II.2.3. Implications of the Geometric Conservation Law

From the analysis presented in the previous section, it follows that an appropriate scheme for evaluating $\int_{t^n}^{t^{n+1}} F_i(\mathbf{w}^*, \mathbf{x}, \dot{\mathbf{x}})dt$ is a scheme that respects the GCL condition (20). Note that once a mesh updating scheme is given, the left hand side of Eq. (20) can be always computed. Hence, a proper method for evaluating $\int_{t^n}^{t^{n+1}} F_i(\mathbf{w}^*, \mathbf{x}, \dot{\mathbf{x}})dt$ is one that obeys the GCL and therefore computes exactly the right hand side of Eq. (20) — that is, $\int_{t^n}^{t^{n+1}} \int_{\partial C_i(\mathbf{x})} \dot{x}\, d\sigma\, dt$.

In three-dimensional space each tetrahedral cell is bounded by a collection of triangular facets. Let $I_{[abc]}$ denote the mesh velocity flux crossing a facet $[abc]$:

$$I_{[abc]} = \int_{t^n}^{t^{n+1}} \int_{[abc]} \dot{x}.\vec{n}\, d\sigma dt \tag{21}$$

and let x_a, x_b and x_c denote the instantaneous positions of the three connected vertices a, b and c. The position of any point on the facet can be parametrized as follows (see Figure II.2)

$$\begin{aligned} x &= \alpha_1 \mathbf{x}_a(t) + \alpha_2 \mathbf{x}_b(t) + (1 - \alpha_1 - \alpha_2)\mathbf{x}_c(t) \\ \dot{x} &= \alpha_1 \dot{\mathbf{x}}_a(t) + \alpha_2 \dot{\mathbf{x}}_b(t) + (1 - \alpha_1 - \alpha_2)\dot{\mathbf{x}}_c(t) \\ \alpha_1 &\in [0,\ 1];\quad \alpha_2 \in [0,\ 1 - \alpha_1];\quad t \in [t^n,\ t^{n+1}] \end{aligned} \tag{22}$$

where

$$x_i(t) = \delta(t)x_i^{n+1} + (1 - \delta(t))x_i^n \quad i = a,\ b,\ c \tag{23}$$

and $\delta(t)$ is a real function satisfying

$$\delta(t^n) = 0;\quad \delta(t^{n+1}) = 1 \tag{24}$$

Substituting the above parametrization in Eq. (21) leads to

$$I_{[abc]} = \int_0^1 \frac{1}{3}(\Delta x_a + \Delta x_b + \Delta x_c).(x_{ac} \wedge x_{bc})\, d\delta \tag{25}$$

where

$$\begin{aligned} x_{ac} = x_a - x_c; \quad x_{bc} = x_b - x_c; \quad \Delta x_a = x_a^{n+1} - x_a^n \\ \Delta x_b = x_b^{n+1} - x_b^n; \quad \Delta x_c = x_c^{n+1} - x_c^n \end{aligned} \tag{26}$$

and the mesh velocities $\dot{x}_a$, $\dot{x}_b$ and $\dot{x}_c$ are obtained from the differentiation of Eqs. (23):

$$\dot{x}_a = \dot{\delta}(t)(x_a^{n+1} - x_a^n) \quad i = a,\ b,\ c \tag{27}$$

Finally, noting that

$$x_{ac} \wedge x_{bc} = \left((\delta x_{ac}^{n+1} + (1-\delta)x_{ac}^n) \wedge (\delta x_{bc}^{n+1} + (1-\delta)x_{bc}^n)\right) \tag{28}$$

is a quadratic function of δ, it becomes clear that the integrand of $I_{[abc]}$ is quadratic in δ, and therefore can be exactly computed using a *two-point integration rule*, provided that Eqs. (27) hold. That is,

$$\dot{x} = \dot{\delta}(t)(x^{n+1} - x^n) = \frac{\Delta\delta}{\Delta t}(x^{n+1} - x^n) \tag{29}$$

which in view of Eq. (24) can also be written as:

$$\boxed{\dot{x} = \frac{x^{n+1} - x^n}{\Delta t}} \tag{30}$$

Summarizing, an appropriate method for evaluating $\int_{t^n}^{t^{n+1}} F_i(\mathbf{w}, \mathbf{x}, \dot{\mathbf{x}})\, dt$ that respects the GCL condition (20) is

$$\boxed{\begin{aligned} \int_{t^n}^{t^{n+1}} F_i(\mathbf{w}, \mathbf{x}, \dot{\mathbf{x}})\, dt &= \frac{\Delta t}{2}\left(F_i(\mathbf{w}^n, \mathbf{x}^{m1}, \dot{\mathbf{x}}^{n+\frac{1}{2}})\right. \\ &\quad \left. + F_i(\mathbf{w}^n, \mathbf{x}^{m2}, \dot{\mathbf{x}}^{n+\frac{1}{2}})\right) \\ m1 &= n + \frac{1}{2} + \frac{1}{2\sqrt{3}} \\ m2 &= n + \frac{1}{2} - \frac{1}{2\sqrt{3}} \\ \mathbf{w}^{n+\eta} &= \eta\mathbf{w}^{n+1} + (1-\eta)\mathbf{w}^n \\ \mathbf{x}^{n+\eta} &= \eta\mathbf{x}^{n+1} + (1-\eta)\mathbf{x}^n \\ \dot{\mathbf{x}}^{n+\frac{1}{2}} &= \frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{2} \end{aligned}} \tag{31}$$

II.3. A STAGGERED EXPLICIT/IMPLICIT TIME INTEGRATOR

II.3.1. Background

When the structure undergoes small displacements, the fluid mesh can be frozen and "transpiration" fluxes can be introduced at the fluid side of the fluid/structure boundary to account for the motion of the structure. In that case, the transient aeroelastic problem is simplified from a three- to a two-field coupled problem.

Furthermore, if the structure is assumed to remain in the linear regime and the fluid flow is linearized around an equilibrium position W_0 (note that most fluid/structure instability problems are analyzed by investigating the response of the coupled system to a perturbation around a steady state), the semi-discrete equations governing the coupled aeroelastic problem become (see [33] for details):

$$\begin{aligned} \begin{bmatrix} \dot{\delta\mathbf{w}} \\ \dot{\mathbf{s}} \end{bmatrix} &= \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{E} \end{bmatrix} \begin{pmatrix} \delta\mathbf{w} \\ \mathbf{s} \end{pmatrix} \\ \begin{bmatrix} \delta\mathbf{w} \\ \mathbf{s} \end{bmatrix}_{(t=0)} &= \begin{bmatrix} \delta\mathbf{w} \\ \mathbf{s} \end{bmatrix}_0 \end{aligned} \tag{32}$$

where $\delta\mathbf{w}$ is the perturbed fluid state vector, $\mathbf{s} = \begin{pmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{pmatrix}$ is the structure state vector, matrix $\mathbf{A}$ results from the spatial discretization of the flow equations, $\mathbf{B}$ is the matrix induced by the transpiration fluxes at the fluid/structure boundary $\Gamma_{F/S}$, $\mathbf{C}$ is the matrix that transforms the fluid pressure on $\Gamma_{F/S}$ into prescribed structural forces; finally $\mathbf{E} = \begin{bmatrix} \mathbf{O} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{D} \end{bmatrix}$, where $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{K}$ are the structural mass, damping, and stiffness matrices.

A mathematical discussion of the time-integration of Eqs. (32) via implicit/implicit and explicit/implicit partitioned procedures can be found in Ref. [33]. In the present work we focus on the more general three-way coupled aeroelastic problem (1). Based on the mathematical results established by Farhat, Fezoui and Lanteri [12] for solving Eqs. (32), we design a family of explicit/implicit staggered procedures for time-integrating Eqs. (1), and address important issues pertaining to accuracy, stability, distributed computing, subcycling, and parallel processing.

II.3.2. A0: An Explicit/Implicit Algorithm

We consider 3D nonlinear Euler flows and linear structural vibrations. From the results established in Section II.2, it follows that the semi-discrete equations governing the three-way coupled aeroelastic problem can be written as:

$$
\begin{aligned}
&\mathbf{V}(\mathbf{x}^{n+1})\mathbf{w}^{n+1} - \mathbf{V}(\mathbf{x}^{n})\mathbf{w}^{n} \\
&\qquad + \frac{\Delta t}{2}(\mathbf{F}^{c}(\mathbf{w}^{n},\ \mathbf{x}^{m1},\ \dot{\mathbf{x}}^{n+\frac{1}{2}}) \\
&\qquad + \mathbf{F}^{c}(\mathbf{w}^{n},\ \mathbf{x}^{m2},\ \dot{\mathbf{x}}^{n+\frac{1}{2}})) \ = \ 0 \\
&\qquad m1 = n + \frac{1}{2} + \frac{1}{2\sqrt{3}} \\
&\qquad m2 = n + \frac{1}{2} - \frac{1}{2\sqrt{3}} \\
&\qquad \mathbf{w}^{n+\eta} = \eta\mathbf{w}^{n+1} + (1-\eta)\mathbf{w}^{n} \\
&\qquad \mathbf{x}^{n+\eta} = \eta\mathbf{x}^{n+1} + (1-\eta)\mathbf{x}^{n} \\
&\qquad \dot{\mathbf{x}}^{n+\frac{1}{2}} \ = \ \frac{\mathbf{x}^{n+1} - \mathbf{x}^{n}}{\Delta t} \\
&\mathbf{M}\ddot{\mathbf{q}}^{n+1} + \mathbf{D}\dot{\mathbf{q}}^{n+1} + \mathbf{K}\mathbf{q}^{n+1} \ = \ \mathbf{f}^{ext}(\mathbf{w}^{n+1}(\mathbf{x}, t)) \\
&\widetilde{\mathbf{M}}\ddot{\mathbf{x}}^{n+1} + \widetilde{\mathbf{D}}\dot{\mathbf{x}}^{n+1} + \widetilde{\mathbf{K}}\mathbf{x}^{n+1} \ = \ \mathbf{K}_{c}\,\mathbf{q}^{n+1}
\end{aligned}
\tag{33}
$$

In many aeroelastic problems such as flutter analysis, a steady flow is first computed around a structure in equilibrium. Next, the structure is perturbed via an initial displacement and/or velocity and the aeroelastic response of the coupled fluid/structure system is analyzed. This suggests that a natural sequencing for the staggered time-integration of Eqs. (33) is:

1. Perturb the structure via some initial conditions.

2. Update the fluid grid to conform to the new structural boundary.

3. Advance the flow with the new boundary conditions.

4. Advance the structure with the new pressure load.

5. Repeat from step (2) until the goal of the simulation (flutter detection or suppression) is reached.

An important feature of partitioned solution procedures is that they allow the use of existing single discipline software modules. In this effort, we are particularly interested in re-using a 3D version of the massively parallel explicit flow solver described by Farhat, Lanteri and Fezoui [10,12,14,25].

Therefore, we select to time-integrate the semi-discrete fluid equations with a 3-step variant of the explicit Runge-Kutta algorithm. On the other hand, the aeroelastic response of a structure is often dominated by low frequency dynamics. Hence, the structural equations are most efficiently solved by an implicit time-integration scheme. Here, we select to time-integrate the structural motion with the implicit midpoint rule (IMR) because it allows enforcing both continuity Eqs. (2) while still respecting the GCL condition (see [26]). Consequently, we propose the following explicit/implicit solution algorithm for solving the three-field coupled problem (33):

1. Update the fluid grid:

$$\textit{Solve} \quad \widetilde{\mathbf{M}}\ddot{\mathbf{x}}^{n+1} + \widetilde{\mathbf{D}}\dot{\mathbf{x}}^{n+1} + \widetilde{\mathbf{K}}\mathbf{x}^{n+1} = \mathbf{K}_c\,\mathbf{q}^n$$

$$\textit{Compute} \quad \mathbf{x}^{m1},\ \mathbf{x}^{m2} \quad \text{from Eq. (33)}$$

$$\dot{\mathbf{x}}^{n+\frac{1}{2}} = \frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{\Delta t}$$

2. Advance the fluid system using RK3:

$$W_i^{n+1^{(0)}} = W_i^n$$

$$W_i^{n+1^{(k)}} = \frac{V_i(\mathbf{x}^n)}{V_i(\mathbf{x}^{n+1})} W_i^{n+1^{(0)}} + \frac{1}{V_i(\mathbf{x}^{n+1})} \frac{1}{4-k} \frac{\Delta t}{2} \left(F_i(\mathbf{w}^n,\ \mathbf{x}^{m1},\ \dot{\mathbf{x}}^{n+\frac{1}{2}}) + F_i(\mathbf{w}^n,\ \mathbf{x}^{m2},\ \dot{\mathbf{x}}^{n+\frac{1}{2}}) \right) \quad k = 1,\ 2,\ 3 \tag{34}$$

$$W_i^{n+1} = W_i^{n+1^{(3)}}$$

3. Advance the structural system using IMR:

$$\mathbf{M}\ddot{\mathbf{q}}^{n+1} + \mathbf{D}\dot{\mathbf{q}}^{n+1} + \mathbf{K}\mathbf{q}^{n+1} = \mathbf{f}^{ext}(\mathbf{w}^{n+1})$$

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \frac{\Delta t}{2}(\dot{\mathbf{q}}^n + \dot{\mathbf{q}}^{n+1})$$

$$\dot{\mathbf{q}}^{n+1} = \dot{\mathbf{q}}^n + \frac{\Delta t}{2}(\ddot{\mathbf{q}}^n + \ddot{\mathbf{q}}^{n+1})$$

In the sequel the above explicit/implicit staggered procedure is referred to as A0. It is depicted in Figure II.3. Extensive experiments with this solution procedure have shown that its stability limit is governed by the critical time-step of the explicit fluid solver (and therefore is not worse than that of the underlying fluid explicit time-integrator).

The 3-step Runge-Kutta algorithm is third-order accurate for linear problems and second-order accurate for nonlinear ones. The midpoint rule is second-order accurate. A simple Taylor expansion shows that the partitioned procedure A0 is first-order accurate when applied to the linearized Eqs. (32). When applied to Eqs. (33), its accuracy depends on the solution scheme selected for solving the fluid mesh equations of motion. As long as the time-integrator applied to these equations is consistent, A0 is guaranteed to be at least first-order accurate.

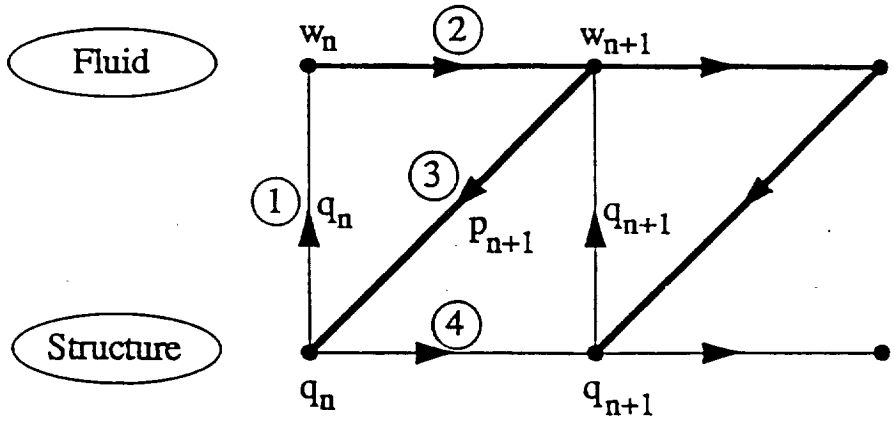


Figure II.3. The basic FSI staggered procedure A0.

II.4. SUBCYCLING

The fluid and structure fields have often different physical time scales. For problems in aeroelasticity, the fluid flow usually requires a smaller temporal resolution than the structural vibration. Therefore, if A0 is used to solve Eqs. (33), the coupling time-step Δt_c will be typically dictated by the stability time-step of the fluid system Δt_F and not the time-step $\Delta t_S > \Delta t_F$ that meets the accuracy requirements of the structural field.

Subcycling the fluid computations with a factor $n_{S/F} = \Delta t_S / \Delta t_F$ can offer substantial computational advantages, including

- savings in the overall simulation CPU time, because in that case the structural field will be advanced fewer times.

- savings in I/O transfers and/or communication costs when computing on a heterogeneous platform, because in that case the fluid and structure kernels will exchange information fewer times.

However, these advantages are effectively realized only if subcycling does not restrict the stability region of the staggered algorithm to values of the coupling time-step Δt_c that are small enough to offset these advantages. In Ref. [33] it is shown that for the linearized problem (32), the straight forward conventional subcycling procedure — that is, the scheme where at the end of each $n_{S/F}$ fluid subcycles only the interface pressure computed during the last fluid subcycle is transmitted to the structure — lowers the stability limit of A0 to a value that is less than the critical time-step of the fluid explicit time-integrator.

On the other hand, it is also shown in Ref. [33] that when solving Eqs. (32), the stability limit of A0 can be preserved if: (a) the deformation of the fluid mesh between t^n and t^{n+1} is evenly distributed among the $n_{S/F}$ subcycles, and (b) at the end of each $n_{S/F}$ fluid subcycles, the average of the

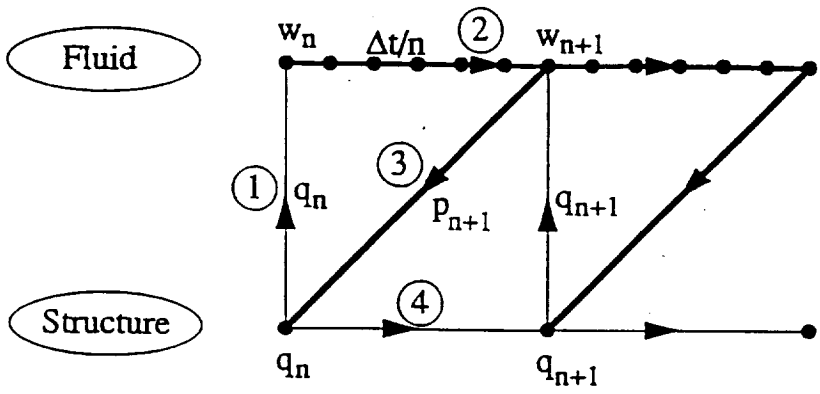


Figure II.4. The fluid-subcycled staggered algorithm A1.

interface pressure field $\overline{p}_{\Gamma_{F/S}}$ computed during the subcycles between t^n and t^{n+1} is transmitted to the structure. Hence, a generalization of A0 is the explicit/implicit fluid-subcycled partitioned procedure depicted in Figure II.4 for solving Eqs. (33). This algorithm is denoted by A1.

Extensive numerical experiments have shown that for small values of $n_{S/F}$, the stability limit of A1 is governed by the critical time-step of the explicit fluid solver. However, experience has also shown that there exists a maximum subcycling factor beyond which A1 becomes numerically unstable. From the theory developed in [12] for the linearized Eqs. (32), it follows that A1 is first-order accurate, and that as one would have expected, subcycling amplifies the fluid errors by the factor $n_{S/F}$.

II.5. EXPLOITING INTER-FIELD PARALLELISM

Both algorithms A0 and A1 are inherently sequential. In both of these procedures, the fluid system must be updated before the structural system can be advanced. Of course, A0 and A1 allow intra-field parallelism (parallel computations within each system), but they inhibit inter-field parallelism. Advancing the fluid and structural systems simultaneously is appealing because it can reduce the total simulation time.

A simple variant of A1 (or A0 if subcycling is not desired) that allows inter-field parallel processing is graphically shown in Figure II.5. This variant is called A2.

Using A2, the fluid and structure kernels can run in parallel during the time-interval $[t_n, t_{n+n_{S/F}}]$. Inter-field communication or I/O transfer is needed only at the beginning of each time-interval. The theory developed in Ref. [33] shows that for the linearized Eqs. (32), A2 is first-order accurate, but parallelism is achieved at the expense of amplified errors in the fluid and structure responses.

In order to improve the accuracy of the basic parallel time-integrator A2, we have investigated

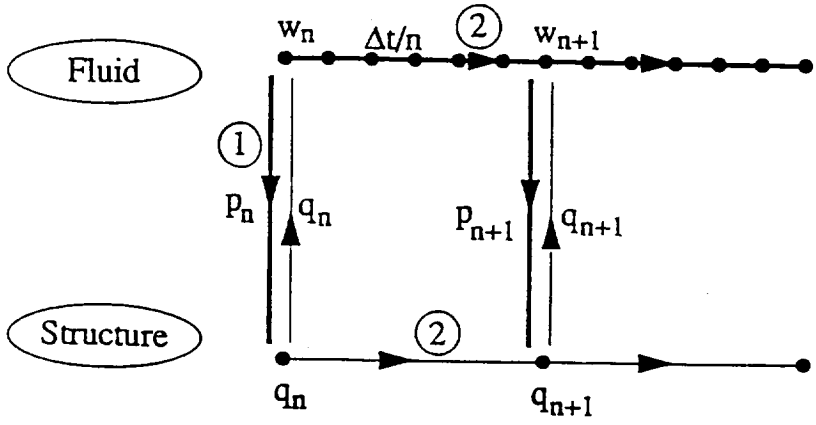


Figure II.5. The inter-parallel, fluid-subcycled staggered algorithm A2.

exchanging information between the fluid and structure kernels at half-steps as illustrated in Figure II.6. The resulting algorithm is called A3.

For algorithm A3, the first half of the computations is identical to that of A2, except that the fluid system is subcycled only up to $t^{n+\frac{n_{S/F}}{2}}$, while the structure is advanced in one step up to $t^{n+n_{S/F}}$. At the time $t^{n+\frac{n_{S/F}}{2}}$, the fluid and structure kernels exchange pressure, displacement and velocity information. In the second-half of the computations, the fluid system is subcycled from $t^{n+\frac{n_{S/F}}{2}}$ to $t^{n+n_{S/F}}$ using the new structural information, and the structural behavior is re-computed in parallel using the newly received pressure distribution. Note that the first evaluation of the structural state

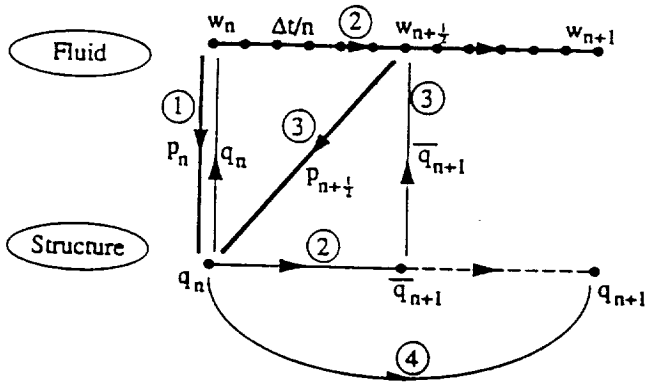


Figure II.6. The midpoint-corrected, inter-parallel, fluid-subcycled staggered algorithm A3.

vector can be interpreted as a predictor.

It can be shown that when applied to the linearized Eqs. (32), A3 is first-order accurate and reduces the errors of A2 by the factor $n_{S/F}$, at the expense of one additional communication step or I/O transfer during each coupled cycle (see [12] for a detailed error analysis).

II.6. COMPUTER IMPLEMENTATION ISSUES

II.6.1. Incompatible mesh interfaces

In general, the fluid and structure meshes have two independent representations of the physical fluid/structure interface. When these representations are identical — for example, when every fluid grid point on $\Gamma_{F/S}$ is also a structural node and *vice-versa* — the evaluation of the pressure forces and the transfer of the structural motion to the fluid mesh are trivial operations. However, analysts usually prefer to be free of such restrictions. In particular:

- Be able to use a fluid mesh and a structural model that have been independently designed and validated.

- Be able to refine each mesh independently from the other.

Hence, most realistic aeroelastic simulations will involve handling fluid and structural meshes that are incompatible at their interface boundaries (Figure II.7). In Ref. [29], we have addressed this issue and proposed a preprocessing "matching" procedure that does not introduce any other

approximation than those intrinsic to the fluid and structure solution methods. This procedure can be summarized as follows.

The nodal forces induced by the fluid pressure on the "wet" surface of a structural element e can be written as:

$$f_i = -\int_{\overline{\Omega}^{(e)}} N_i p v \, d\sigma \tag{35}$$

where $\overline{\Omega}^{(e)}$ denotes the geometrical support of the wet surface of the structural element e, p is the pressure field, v is the unit normal to $\overline{\Omega}^{(e)}$, and N_i is the shape function associated with node i. Most if not all FE structural codes evaluate the integral in Eq. (35) via a quadrature rule:

$$f_i = -\sum_{g=1}^{g=n_g} w_g N_i(X_g) p(X_g) \tag{36}$$

where w_g is the weight of the Gauss point X_g. Hence, a structural code needs to know the values of the pressure field only at the Gauss points of its wet surface. This information can be easily made available once every Gauss point of a wet structural element is paired with a fluid cell (Figure II.8). It should be noted that in Eq. (36), X_g are not necessarily the same Gauss points as those used for stiffness evaluation. For example, if a high pressure gradient is anticipated over a certain wet area of the structure, a larger number of Gauss points can be used for the evaluation of the pressure forces f_i on that area.

On the other hand, when the structure moves and/or deforms, the motion of the fluid grid points on $\Gamma_{F/S}$ can be prescribed via the regular FE interpolation:

$$\mathbf{x}(S_j) = \sum_{k=1}^{k=wne} N_k(\mathcal{X}_j)\mathbf{q}_k^{(e)} \tag{37}$$

where S_j, wne, $\mathcal{X}_j$, and $\mathbf{q}_k$ denote respectively a fluid grid point on $\Gamma_{F/S}$, the number of wet nodes in its "nearest" structural element e, the natural coordinates of S_j in $\overline{\Omega}^{(e)}$, and the structural displacement at the k-th node of element e. From Eq. (37), it follows that each fluid grid point on $\Gamma_{F/S}$ must be matched with one wet structural element (see Figure II.9).

Given a fluid grid and a structural model, constructed independently, the Matcher program described in Ref. [29] generates all the data structures needed to evaluate the quantities in Eqs. (39,40) in a single preprocessing step.

II.6.3. Intra-field parallel processing

Aeroelastic simulations are known to be computationally intensive and therefore can benefit from the parallel processing technology. An important feature of a partitioned solution procedure is preservation of software modularity. Hence, all of the solution procedures A0, A1, A2 and A3 can use existing computational fluid dynamics and computational structural mechanics parallel algorithms. The solution of the mesh motion equations can be easily incorporated into an existing fluid code, and its parallelization is not more difficult than that of a FE structural algorithm.

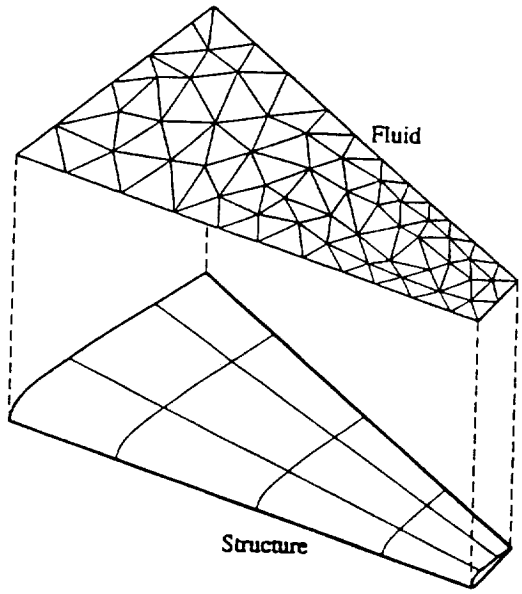


Figure II.7. Mismatched fluid-structure discrete interfaces.

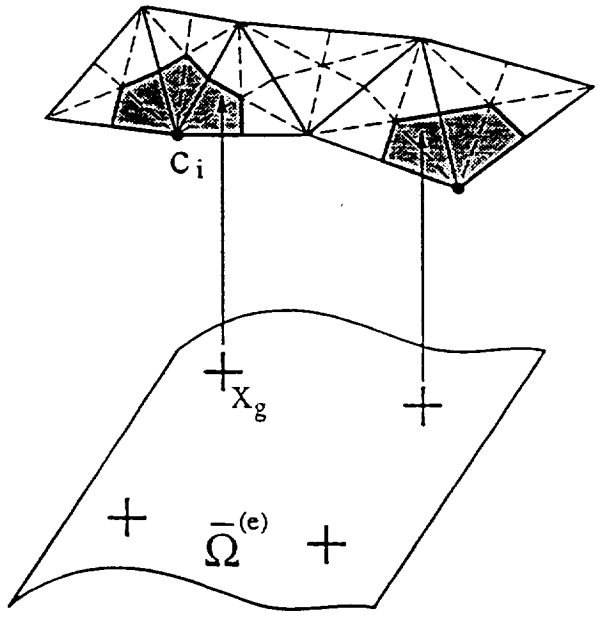


Figure II.8. Pairing of structural Gauss points and fluid cells.

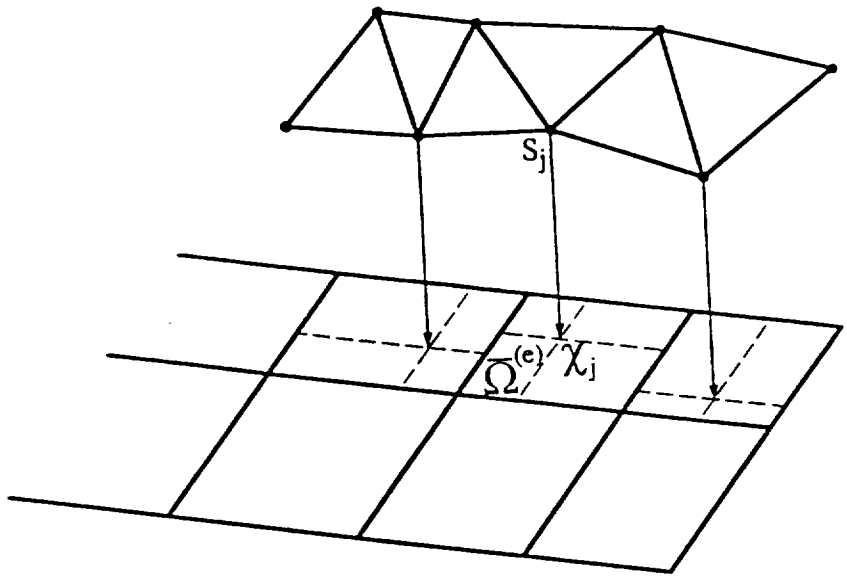


Figure II.9. Pairing of fluid point and wet structural element

Our approach to parallel processing is based on the mesh partitioning/message-passing paradigm, which leads to a portable software design. Using an automatic mesh partitioning algorithm [13,17] both fluid and structural meshes are decomposed into subdomains. The same "old" serial code is executed within every subdomain. The "gluing" or assembly of the subdomain results is implemented in a separate software module.

This approach enforces data locality [25] and is therefore suitable for all parallel hardware architectures. Note that in this context, message-passing refers to the assembly phase of the subdomain results. It does not imply that messages have to be explicitly exchanged between the subdomains. For example, message-passing can be implemented on a shared memory multiprocessor as a simple access to a shared buffer, or as a duplication of one buffer into another.

II.6.4. Inter-field parallel processing

Using the message-passing paradigm, inter-field parallel processing can be implemented in the same manner as intra-field multiprocessing. The fluid and structure codes can run either on different sequential or parallel machines, or on a different partition of the same multiprocessor. Any software product such as PVM [21] can be used to implement message-passing between the two computational kernels.

II.7. APPLICATIONS AND PRELIMINARY RESULTS

II.7.1. Transonic Wing Benchmark (3D)

Here we illustrate the aeroelastic computational methodology described in the previous sections

Table II.1 Characteristics of the fluid meshes M1–M4 for 3D benchmark

Mesh	N_{var}	N_{tet}	N_{fac}	N_{var}
M1	15460	80424	99891	77300
M2	31513	161830	201479	157565
M3	63917	337604	415266	319585
M4	115351	643392	774774	576755

with some preliminary numerical investigations on an iPSC-860, a Paragon XP/S, a Cray T3D, and an IBM SP2 massively parallel systems, of the aerodynamics and aeroelastic transient response of a 3D wing in a transonic airstream.

The wing is represented by an equivalent plate model discretized by 1071 triangular plate elements, 582 nodes, and 6426 degrees of freedom (Figure II.10). Four meshes identified as M1 through M4, are designed for the discretization of the 3D flow domain around the wing. The characteristics of theses meshes are given in Table II.1 where N_{var}, N_{tet}, N_{fac}, and N_{var} denote respectively the number of vertices, tetrahedra, facets (edges), and fluid variables, respectively. A partial view of the discretization of the flow domain is shown in Figure II.11.

The sizes of the fluid meshes M1–M4 have been tailored for parallel computations on respectively 16 (M1), 32 (M2), 64 (M3), and 128 processors (M4) of a Paragon XP/S and a Cray T3D systems. In particular, the sizes of these meshes are such that the processors of a Paragon XP/S machine with 32 Mbytes per node would not swap when solving the corresponding flow problems.

Because the fluid and structural meshes are not compatible at their interface (Figure II.12), the Matcher software [29] is used to generate in a single preprocessing step the data structures required for transferring the pressure load to the structure, and the structural deformations to the fluid.

II.7.2. The Flow Solver and its Parallelization

The Euler flow equations are solved with a second-order accurate FV Monotonic Upwinding Scheme for Conservation Laws (MUSCL) [40,30] on fully unstructured grids. The resulting semi-discrete equations are time-integrated using a second-order low-storage explicit Runge-Kutta method. Further details regarding this explicit unstructured flow solver and its subdomain based parallelization can be found in recent publications [10,12,14,25].

In this work, the unstructured dynamic fluid mesh is represented by the pseudo-structural model of Batina [3] ($\widetilde{\mathbf{M}} = \widetilde{\mathbf{D}} = 0$). The grid points located on the upstream and downstream boundaries are held fixed. The motion of those points located on $\Gamma_{F/S}$ is determined from the wing surface motion and/or deformation. At each time-step t^{n+1}, the new position of the interior grid points is determined from the solution of the displacement driven pseudo-structural problem via the two-step

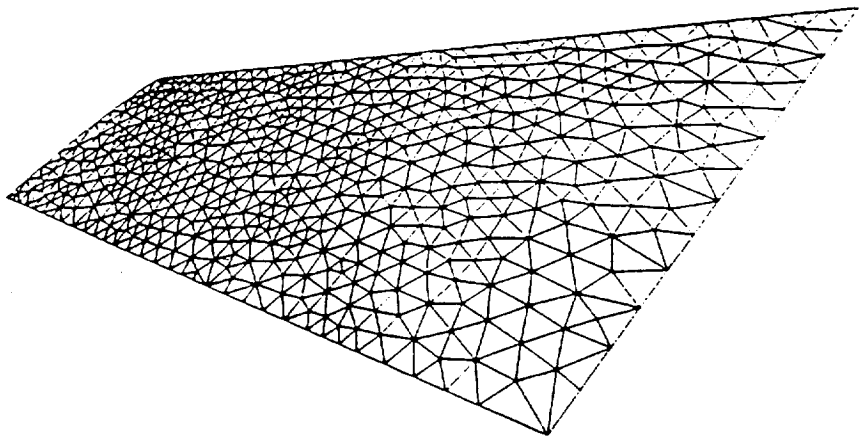


Figure II.10. The discrete structural model.

iterative procedure described in [14].

II.7.3. The Parallel Structure Solver

The structural equations of dynamic equilibrium are solved with the parallel implicit transient Finite Element Tearing and Interconnecting (FETI) method [15]. Because it is based on a midpoint rule formulation, this method allows enforcing both continuity Eqs. (2) while still respecting the GCL condition. The resistance of the structure to displacements in the plane of the skin is assumed to be small. Consequently, all structural computations are performed with a linearized structural theory. Since the FETI solver is a domain decomposition based iterative solver, we also use the special restarting procedure proposed in Ref. [16] for the efficient iterative solution of linear systems with repeated right hand sides.

II.7.4. Computational Platforms

Computations were performed on the following massively parallel computers: Intel iPSC-860 hypercube, Intel Paragon XP/S, Cray T3D, and IBM SP2, using double precision floating-point arithmetic throughout. Message passing is carried out via NX on the Paragon XP/S multiprocessor, PVM T3D on the Cray T3D system, and MPI on the IBM SP2. On the hypercube, fluid and structure solvers are implemented as separate programs that communicate via the intercube communication procedures described by Barszcz [2].

II.7.5. Performance of the Parallel Flow Solver

The performance of the parallel flow solver is assessed with the computation of the steady state of a flow around the given wing at a Mach number $M_\infty = 0.84$ and an angle of attack $\beta = 3.06$ degrees. The CFL number is set to 0.9. The four meshes M1–M4 are decomposed in respectively 16, 32, 64, and 128 *overlapping* subdomains using the mesh partitioner described in [28]. The

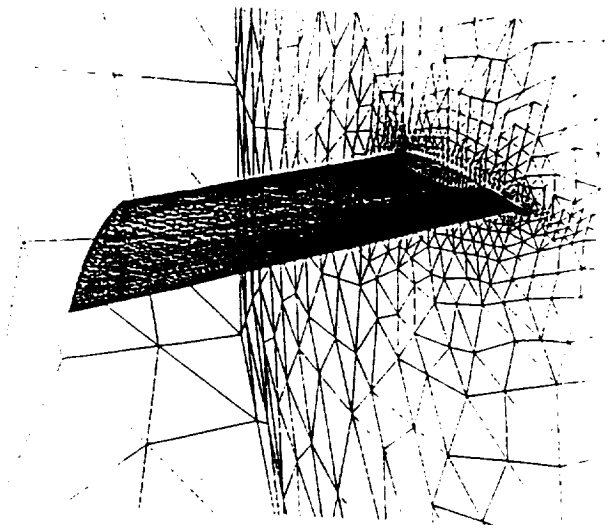


Figure II.11. The discrete flow domain (partial view).

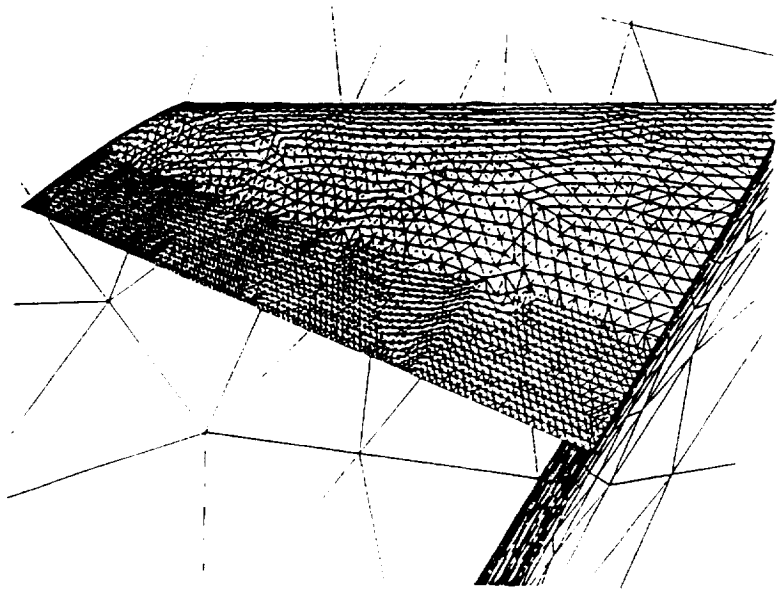


Figure II.12. Fluid/structure interface incompatibilities

motivation for employing overlapping subdomains and the impact of this computational strategy on parallel performance are discussed in Ref. [14]. Measured times in seconds are reported in Tables II.2 through II.4 for the first 100 time steps on a Paragon XP/S machine (128 processors), a Cray T3D system (128 processors), and an IBM SP2 computer (128 processors), respectively. In these tables, N_p, N_{var}, T_{comm}^{loc}, T_{comm}^{glo}, T_{comp}, T_{tot} and Mflops denote respectively the number of processors, the number of variables (unknowns) to be solved, the time elapsed in short range interprocessor communication between neighboring subdomains, the time elapsed in long range

Table II.2. Performance of the parallel flow solver on the Paragon XP/S system for 16–128 processors (100 time steps — CFL = 0.9)

Mesh	N_p	N_{var}	T_{comm}^{loc}	T_{comm}^{glo}	T_{comp}	T_{tot}	Mflops
M1	16	77,300	2.0 s.	40.0 s.	96.0 s.	138.0 s.	84
M2	32	157,565	4.5 s.	57.0 s.	98.5 s.	160.0 s.	145
M3	64	319,585	7.0 s.	90.0 s.	103.0 s.	200.0 s.	240
M4	128	576,755	6.0 s.	105.0 s.	114.0 s.	225.0 s.	401

global interprocessor communication, the computational time, the total simulation time, and the computational speed in millions of floating point operations per second. Communication and computational times were not measured separately on the SP2.

Typically, short range communication is needed for assembling various subdomain results such as fluxes at the subdomain interfaces, and long range interprocessor communication is required for reduction operations such as those occurring in the the evaluation of the stability time-steps and the norms of the nonlinear residuals. It should be noted that we use the same fluid code for steady state and aeroelastic computations. Hence, even though we are benchmarking in Tables II.2–II.4 a steady state computation with a local time stepping strategy, we are still timing the kernel that evaluates the global time-step in order to reflect its impact on the unsteady computations that we perform in aeroelastic simulations such as those that are discussed next. The megaflop rates reported in Tables II.2 through II.4 are computed in a conservative manner: they exclude all the redundant computations associated with the overlapping subdomain regions.

Table II.3. Performance of the parallel flow solver on the Cray T3D system for 16–128 processors (100 time steps — CFL = 0.9)

Mesh	N_p	N_{var}	T_{comm}^{loc}	T_{comm}^{glo}	T_{comp}	T_{tot}	Mflops
M1	16	77,300	1.6 s.	2.1 s.	87.3 s.	91.0 s.	127
M2	32	157,565	2.5 s.	4.1 s.	101.4 s.	108.0 s.	215
M3	64	319,585	3.5 s.	7.2 s.	100.3 s.	111.0 s.	433
M4	128	576,755	3.0 s.	7.2 s.	85.3 s.	95.5 s.	945

It may be readily verified that the number of processors assigned to each mesh is such that N_{var}/N_p is almost constant. This means that larger numbers of processors are attributed to larger meshes in order to keep each local problem within a processor at an almost constant size. For such a benchmarking strategy, *parallel scalability* of the flow solver on a target parallel processor implies

Table II.4. Performance of the parallel flow solver on the IBM SP2 system for 16–128 processors (100 time steps — CFL = 0.9)

Mesh	N_p	N_{var}	T_{comm}^{loc}	T_{comm}^{glo}	T_{comp}	T_{tot}	Mflops
M1	16	77,300				10.8 s.	1072
M2	32	157,565				12.0 s.	1930
M3	64	319,585				12.8 s.	3785
M4	128	576,755				11.9 s.	7430

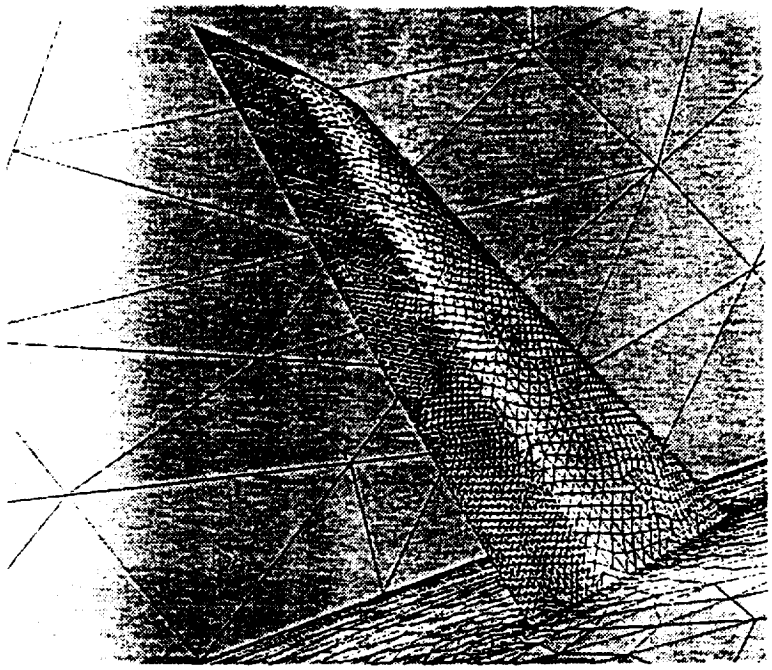


Figure II.13. Mach number isosurfaces for the steady-state regime.

that the total solution CPU time should be constant for all meshes and their corresponding number of processors.

This is clearly not the case for the Paragon XP/S system. On this machine, short range communication is shown to be inexpensive, but long range communication costs are observed to be important. This is due to the latency of the Paragon XP/S parallel processor, which is an order of magnitude slower than that of the Cray T3D system. Another possible source of global communication time increase is the load imbalance between the processors since message passing is also used for synchronization. However, this does not seem to be significant on the T3D and SP2 parallel processors.

On the other hand, parallel scalability is well demonstrated for the Cray T3D and IBM SP2 systems. The results reported in Tables II.3 and II.4 show that all computations using meshes M1–M4 and the corresponding number of processors consume almost the same total amount of CPU time. For

128 processors, the Cray T3D system is shown to be more than twice faster than the Paragon XP/S machine. The difference appears to be strictly in long range communication as the computational time is reported to be almost the same on both machines. However, most impressive is the fact that an IBM SP2 with 32 processors only is shown to be three times faster than a 128-processor Paragon XP/S, and faster than a Cray T3D with 128 processors.

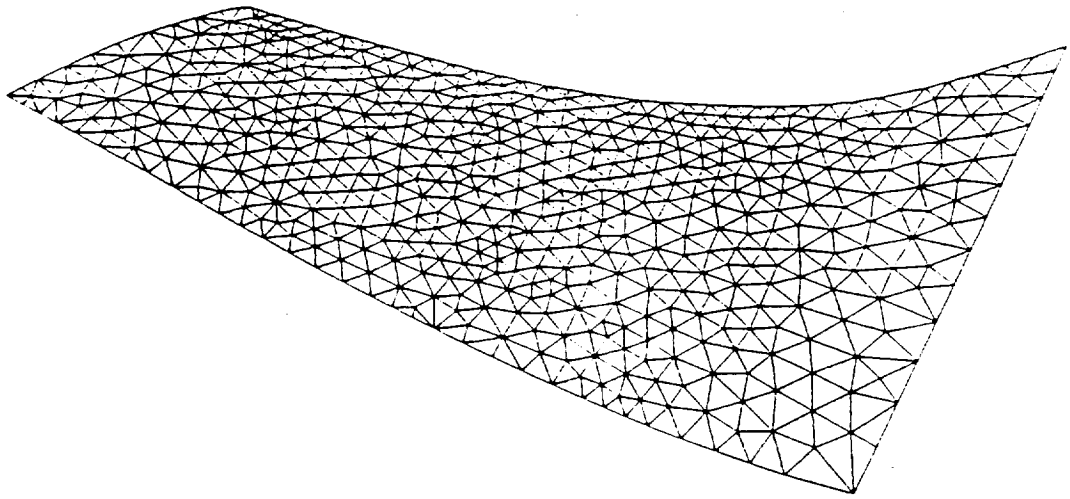


Figure II.14. Initial perturbation of the displacement field of the wing.

II.7.6. Performance of the Parallel Structure Solver

For the performance assessment of the parallel FETI structural solver, we refer the reader to the recent publications [15,16].

II.7.7. Performance of the Partitioned Procedures A0–A3

In order to illustrate the relative merits of the partitioned procedures A0, A1, A2 and A3, we consider first two different series of transient aeroelastic simulations at Mach number $M_\infty = 0.84$ that highlight

- the relative accuracy of these coupled solution algorithms for a fixed subcycling factor $n_{S/F}$.
- the relative speed of these coupled solution algorithms for a fixed level of accuracy.

In all cases, mesh $M2$ is used for the flow computations, 32 processors of an iPSC-860 system are allocated to the fluid solver, and 4 processors of the same machine are assigned to the structural code. Initially, a steady-state flow is computed around the wing at $M_\infty = 0.84$ (Figure II.13), a

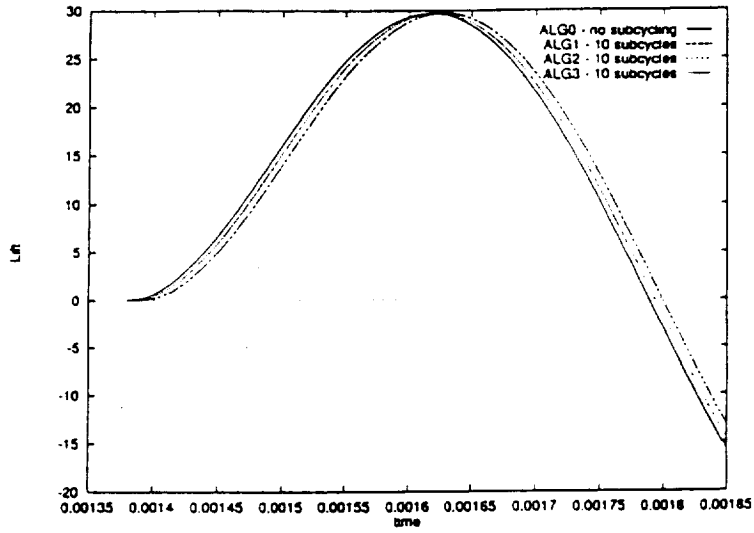


Figure II.15. Lift history for the first half cycle, $n_{S/F} = 10$

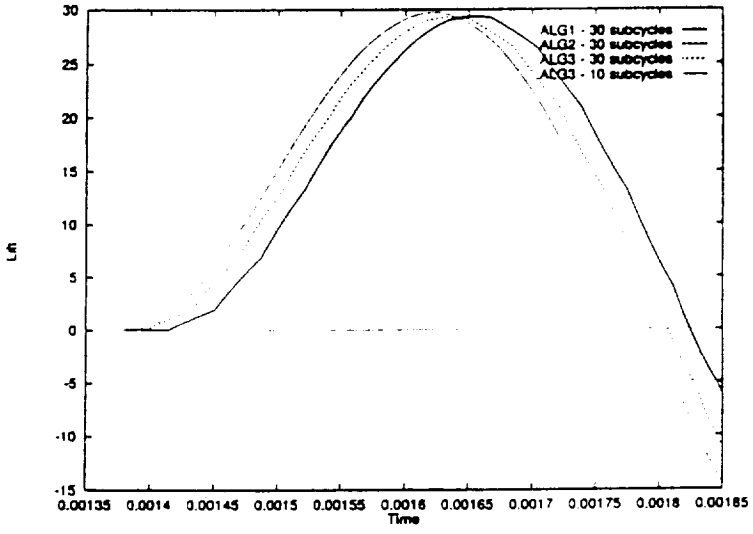


Figure II.16. Lift history for the first half cycle, $n_{S/F} = 30$

Mach number at which the wing described above is not supposed to flutter. Then, the aeroelastic response of the coupled system is triggered by a displacement perturbation of the wing along its first mode (Figure II.14).

First, the subcycling factor is fixed to $n_{S/F} = 10$ then to $n_{S/F} = 30$, and the lift is computed using a time-step corresponding to the stability limit of the explicit flow solver in the absence of coupling

Table II.5. Performance results for coupled FSI problem on the Intel iPSC-860
Fluid: 32 processors, Structure: 4 processors
Elapsed time for 50 fluid time-steps

Alg.	Fluid Solver	Fluid Motion	Struc. Solver	ICWS	ICWF	Total CPU
A0	177.4 s.	71.2 s.	33.4 s.	219.0 s.	384.1 s.	632.7 s.
A1	180.0 s.	71.2 s.	16.9 s.	216.9 s.	89.3 s.	340.5 s.
A2	184.8 s.	71.2 s.	16.6 s.	114.0 s.	0.4 s.	256.4 s.
A3	176.1 s.	71.2 s.	10.4 s.	112.3 s.	0.4 s.	247.7 s.

with the structure. The obtained results are depicted in Figure II.15 and Figure II.16 for the first half cycle.

The superiority of the parallel fluid-subcycled A3 solution procedure is clearly demonstrated in Figure II.15 and Figure II.16. For $n_{S/F} = 10$, A3 is shown to be the closest to A0, which is supposed to be the most accurate since it is sequential and non-subcycled. A1 and A2 have comparable accuracies. However, both of these algorithms exhibit a significantly more important phase error than A3, especially for $n_{S/F} = 30$.

Next, the relative speed of the partitioned solution procedures is assessed by comparing their CPU time for a certain level of accuracy dictated by A0. For this problem, it turned out that in order to meet the accuracy requirements of A0, the solution algorithms A1 and A2 can subcycle only up to $n_{S/F} = 5$, while A3 can easily use a subcycling factor as large as $n_{S/F} = 10$. The performance results measured on an iPSC-860 system are reported on Table II.5 for the first 50 coupled time-steps. In this table, ICWF and ICWS denote the inter-code communication timings measured respectively in the fluid and structural kernels; these timings include idle and synchronization (wait) time when the fluid and structural communications do not completely overlap. For programming reasons, ICWS is monitored together with the evaluation of the pressure load.

II.8. CONCLUSIONS

From the results reported in Table II.5, the following observations can be made.

- The fluid computations dominate the simulation time. This is partly because the structural model is simple in this case, and a linear elastic behavior is assumed. However, by allocating 32 processors to the fluid kernel and 4 processors to the structure code, a reasonable load balance is shown to be achieved for A0.

- During the first 50 fluid time-steps, the CPU time corresponding to the structural solver does not decrease linearly with the subcycling factor $n_{S/F}$ because of the initial costs of the FETI reorthogonalization procedure designed for the efficient iterative solution of implicit systems with repeated right hand sides [16].

- The effect of subcycling on intercube communication costs is clearly demonstrated. The impact of this effect on the total CPU time is less important for A2 and A3 which feature inter-field parallelism in addition to intra-field multiprocessing, than for A1 which features intra-field parallelism only (note that A1 with $n_{S/F} = 1$ is identical to A0), because the flow solution time is dominating.

- Algorithms A2 and A3 allow a certain amount of overlap between inter-field communications, which reduces intercube communication and idle time on the fluid side to less than 0.001% of the amount corresponding to A0.

- The superiority of A3 over A2 is not clearly demonstrated for this problem because of the simplicity of the structural model and the consequent load unbalance between the fluid and structure computations.

Most importantly, the performance results reported in Table II.5 demonstrate that subcycling and inter-field parallelism are desirable for aeroelastic simulations even when the flow computations dominate the structural ones, because these features can significantly reduce the total simulation time by minimizing the amount of inter-field communications and overlapping them. For the simple problem described herein, the parallel fluid-subcycled A2 and A3 algorithms are more than twice faster than the conventional staggered procedure A0.

Acknowledgments

This work was supported by NASA Langley under Grant NAG-1536427, by NASA Lewis Research Center under Grant NAG3-1425, and by the National Science Foundation under Grant ESC-9217394.

REFERENCES

1. W. K. Anderson, J. L. Thomas and C .L. Rumsey, Extension and application of flux-vector splitting to unsteady calculations on dynamic meshes, AIAA Paper No. 87-1152-CP, 1987.

2. E. Barszcz, Intercube communication on the iPSC/860, Scalable High Performance Computing Conference, Williamsburg, April 26-29, 1992.

3. J. T. Batina, Unsteady Euler airfoil solutions using unstructured dynamic meshes, AIAA Paper No. 89-0115, *AIAA 27th Aerospace Sciences Meeting*, Reno, Nevada, January 9–12, 1989.

4. T. Belytschko, P. Smolenski and W. K. Liu, Stability of multi-time step partitioned integrators for first-order finite element systems, *Comput. Meths. Appl. Mech. Engrg.*, **49** (1985) 281–297.

5. M. Blair, M. H. Williams and T. A. Weisshaar, Time domain simulations of a flexible wing in subsonic compressible flow, AIAA Paper No. 90-1153, *AIAA 8th Applied Aerodynamics Conference*, Portland, Oregon, August 20–22, 1990.

6. C. J. Borland and D. P. Rizzetta, Nonlinear transonic flutter analysis, *AIAA Journal*, **20** (1982) 1606–1615.

7. J. Donea, An arbitrary Lagrangian-Eulerian finite element method for transient fluid-structure interactions, *Comput. Meths. Appl. Mech. Engrg.*, **33** (1982) 689–723.

8. C. Farhat and T. Y. Lin, Transient aeroelastic computations using multiple moving frames of reference, AIAA Paper No. 90-3053, *AIAA 8th Applied Aerodynamics Conference*, Portland, Oregon, August 20–22, 1990.

9. C. Farhat, K. C. Park and Y. D. Pelerin, An unconditionally stable staggered algorithm for transient finite element analysis of coupled thermoelastic problems, *Comput. Meths. Appl. Mech. Engrg.*, **85** (1991) 349–365.

10. C. Farhat, S. Lanteri and L. Fezoui, Mixed finite volume/finite element massively parallel computations: Euler flows, unstructured grids, and upwind approximations, in *Unstructured Scientific Computation on Scalable Multiprocessors*, ed. by P. Mehrotra, J. Saltz, and R. Voigt, MIT Press, (1992) 253–283.

11. C. Farhat and T. Y. Lin, A structure attached corotational fluid grid for transient aeroelastic computations, *AIAA Journal*, **31** (1993) 597–599

12. C. Farhat, L. Fezoui, and S. Lanteri, Two-dimensional viscous flow computations on the Connection Machine: unstructured meshes, upwind schemes, and massively parallel computations, *Comput. Meths. Appl. Mech. Engrg.*, **102** (1993) 61–88.

13. C. Farhat and M. Lesoinne, Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics, *Internat. J. Numer. Meths. Engrg.*, **36**, (1993) 745–764.

14. C. Farhat and S. Lanteri, Simulation of compressible viscous flows on a variety of MPPs: computational algorithms for unstructured dynamic meshes and performance results, *Comput. Meths. Appl. Mech. Engrg.*, **119**, (1994) 35–60.

15. C. Farhat, L. Crivelli and F. X. Roux, A transient FETI methodology for large-scale parallel implicit computations in structural mechanics, *Internat. J. Numer. Meths. Engrg.*, **37**, (1994) 1945–1975.

16. C. Farhat, L. Crivelli and F. X. Roux, Extending substructure based iterative solvers to multiple load and repeated analyses, *Comput. Meths. Appl. Mech. Engrg.*, **117** (1994) 195–209.

17. C. Farhat, S. Lanteri and H. D. Simon, TOP/DOMDEC, A software tool for mesh partitioning and parallel processing, *J. Comput. Sys. Engrg.*, in press.

18. C. Farhat, P. S. Chen and P. Stern, Towards the ultimate iterative substructuring method: combined numerical and parallel scalability, and multiple load cases, *J. Comput. Sys. Engrg.*, in press.

19. C. A. Felippa and K. C. Park, Staggered Transient Analysis Procedures for Coupled Dynamic Systems: Formulation, *Comput. Meths. Appl. Mech. Engrg.*, **24**, (1980) 61–112

20. C. A. Felippa, C. Farhat, P.-S. Chen, U. Gumaste, M. Lesoinne and P. Stern, High performance parallel analysis of coupled problems for aircraft propulsion, Progress Report to NASA LeRC for Period 6/94 through 1/95, Report CU-CAS-95-02, Center for Aerospace Structures, University of Colorado, Boulder, February 1995.

21. A. Geist, A. Beguelin, J. Dongarra, R. Mancheck, V. Sunderam, PVM 3.0 User's Guide and Reference Manual, Technical Report ORNL/TM-12187, Oak Ridge National Laboratory, 1993.

22. G. P. Guruswamy, Time-accurate unsteady aerodynamic and aeroelastic calculations of wings using Euler equations, AIAA Paper No. 88-2281, *AIAA 29th Structures, Structural Dynamics and Materials Conference*, Williamsburg, Virginia, April, 18–20, 1988.

23. W. J. Hesse and N. V. S. Mumford, *Jet Propulsion for Aerospace Applications*, 2nd ed., Pitnam Pubs., New York, N.Y., 1964, Chapter 11.

24. O. A. Kandil and H. A. Chuang, Unsteady vortex-dominated flows around maneuvering wings over a wide range of mach numbers, AIAA Paper No. 88-0317,*AIAA 26th Aerospace Sciences Meeting*, Reno, Nevada, January 11–14, 1988.

25. S. Lanteri and C. Farhat, Viscous flow computations on MPP systems: implementational issues and performance results for unstructured grids, in *Parallel Processing for Scientific Computing*, ed. by R. F. Sincovec *et. al.*, SIAM (1993) 65–70.

26. M. Lesoinne and C. Farhat, Stability analysis of dynamic meshes for transient aeroelastic computations, AIAA Paper No. 93-3325, *11th AIAA Computational Fluid Dynamics Conference*, Orlando, Florida, July 6–9, 1993.

27. M. Lesoinne and C. Farhat, Geometric conservation laws for aeroelastic computations Using unstructured dynamic meshes, AIAA Paper No. 95-1709, 1995.

28. M. Loriot, MS3D: Mesh Splitter for 3D Applications, User's Manual.

29. N. Maman and C. Farhat, Matching fluid and structure meshes for aeroelastic computations: a parallel approach, *Computers & Structures*, in press.

30. B. NKonga and H. Guillard, Godunov type method on non-structured meshes for three-dimensional moving boundary problems, *Comput. Meths. Appl. Mech. Engrg.*, **113**, (1994) 183–204.

31. K. C. Park, C. A. Felippa and J. A. DeRuntz, Stabilization of staggered solution procedures for fluid-dtructure interaction analysis, in *Computational Methods for Fluid-Structure Interaction Problems*, ed.

by T. Belytschko and T. L. Geers, AMD Vol. 26, American Society of Mechanical Engineers, ASME, New York (1977) 95–124

32. K. C. Park and C. A. Felippa, Partitioned analysis of coupled systems, in: Computational Methods for Transient Analysis, T. Belytschko and T. J. R. Hughes, Eds., North-Holland Pub. Co. (1983) 157–219.

33. S. Piperno and C. Farhat, Partitioned procedures for the transient solution of coupled aeroelastic problems, *Comput. Meths. Appl. Mech. Engrg.*, to appear.

34. E. Pramono and S. K. Weeratunga, Aeroelastic computations for wings through direct coupling on distributed-memory MIMD parallel computers, AIAA Paper No. 94-0095, 32nd Aerospace Sciences Meeting, Reno, January 10–13, 1994.

35. R. D. Rausch, J. T. Batina and T. Y. Yang, Euler flutter analysis of airfoils using unstructured dynamic meshes, AIAA Paper No. 89-13834, 30th Structures, Structural Dynamics and Materials Conference, Mobile, Alabama, April 3–5, 1989.

36. V. Shankar and H. Ide, Aeroelastic computations of flexible configurations, *Computers & Structures*, **30** (1988) 15–28.

37. T. W. Strganac and D. T. Mook, Numerical model of unsteady subsonic aeroelastic behavior, *AIAA Journal*, **28** (1990) 903–909.

38. T. Tezduyar, M. Behr and J. Liou, A new strategy for finite element computations involving moving boundaries and interfaces - The deforming spatial domain/space-time procedure: I. The concept and the preliminary numerical tests, *Comput. Meths. Appl. Mech. Engrg.*, **94** (1992) 339–351.

39. P. D. Thomas and C. K. Lombard, Geometric conservation law and its application to flow computations on moving grids, *AIAA Journal*, **17**, (1979) 1030–1037.

40. B. Van Leer, Towards the ultimate conservative difference scheme V: a second-order sequel to Godunov's method, *J. Comp. Phys.*, **32** (1979).

Appendix III

LB: A Program for Load-Balancing Multiblock Grids

Summary

This Appendix describes recent research towards load-balancing the execution of ENG10 on parallel machines. ENG10 is a multiblock-multigrid code developed by Mark Stewart of NYMA Research Inc. to perform axisymmetric aerodynamic analysis of complete turbofan engines taking into account combustion, compression and mixing effects through appropriate circumferential averaging. The load-balancing process is based on an iterative strategy for subdividing and recombining the original grid-blocks that discretize distinct portions of the computational domain. The research work reported here was performed by U. Gumaste under the supervision of Prof. C. A. Felippa.

III.1. INTRODUCTION

III.1.1. Motivation

For efficient parallelization of multiblock-grid codes, the requirement of load balancing demands that the grid be subdivided into subdomains of similar computational requirements, which are assigned to individual processors. Load balancing is desirable in the sense that if the computational load of one or more blocks substantially dominates that of others, processors given the latter must wait until the former complete.

Such "computational bottlenecks" can negate the beneficial effect of parallelization. To give an admittedly extreme example, suppose that the aerodynamic discretization uses up 32 blocks which are assigned to 32 processors, and that one of them takes up 5 times longer to complete than the average of the remaining blocks. Then 31 processors on the average will be idle 80% of the time.

Load balancing is not difficult to achieve for unstructured meshes arising with finite-element or finite-volume discretizations. This is because in such cases one deals with element-level granularity, which can be efficiently treated with well studied domain-decomposition techniques for unstructured meshes. In essence such subdomains are formed by groups of connected elements, and elements may be moved from one domain to another with few "strings attached" other than connectivity.

On the other hand for multiblock-grid discretizations the problem is more difficult and has not, to the writers' knowledge, been investigated in any degree of generality within the context of load balancing. The main difficulty is that blocks cannot be arbitrarily partitioned, for example cell by cell, because discretization constraints enforcing grid topological regularity must be respected.

The following is an outline of the program LB developed at the University of Colorado to perform load-balancing of the multiblock discretization used in the program ENG10. This is a multiblock-multigrid code developed by Mark Stewart of NYMA Research Inc. to perform axisymmetric

aerodynamic analysis of complete turbofan engines taking into account combustion, compression and mixing effects through appropriate circumferential averaging [1].

III.1.2. Requirements and Constraints

Multiblock grids are used to discretize complex geometries. A multiblock grid divides the physical subdomain into topologically rectangular blocks. The grid pertaining to each block is regular (structured). For bladed jet engine geometries, this is achieved by a series of programs also written by Mark Stewart, namely `TOPOS`, `TF` and `MS` [2,3], which function as preprocessors to `ENG10`.

Efficient parallelization requires the computational load to be (nearly) equal among all processors. Usually, depending upon the geometry, the computational sizes of component blocks of a multiblock discretization vary and mapping one to each processor would not naturally ensure load balance. The `LB` program attempts to load balance a given multiblock grid so that the resulting subdivisions of the grid are of similar computational cost.

For re-use of `ENG10` to be possible for the parallel version, it is required that the resulting subdivisions of the original multiblock grid be also regular grids or are collections of blocks, each of which contain regular grids. This imposed the restriction that the number of final subdivisions desired be greater than the number of blocks in the original grid. Thus for most cases, ideally, the number of blocks in the original grid should be 10 to 20, because 32 to 128 processors are normally used in present generation MPPs. `LB` works better when the number of available processor substantially exceeds the original number of blocks.

III.1.3. Multiblock Grid and `MS`

`MS` is a program that, given the domain discretization and blade forces, loss and combustor heating data, etc., interpolates the data onto the grid. This program was used as a basis for `LB` as it possesses the data structures most amenable to the task of load balancing.

Blocks are arranged in a C programming language linked list. Each block possesses a set of segments that are lines joining grid points. The number of segments in each direction (transverse and lateral) determines the computational size of the block. Segments can be of different types depending upon where they are located as follows :

1. *False boundary segments.* These are segments located at the interface between blocks. These are called "false" boundaries as they are not actual physical boundaries in the grid but merely lines across which data has to be exchanged between two adjacent blocks. Knowledge about the false boundaries is essential to determine block connectivity.
2. *Internal and solid boundary segments.* These are segments located at the interface of combustors, blades, etc. Knowledge about the internal and solid boundary segments helps determine the location of blades, combustors and other engine components.
3. *Far-field boundary segments.* These are segments located at the far-field boundaries of the domain and are useful in imposing boundary conditions.

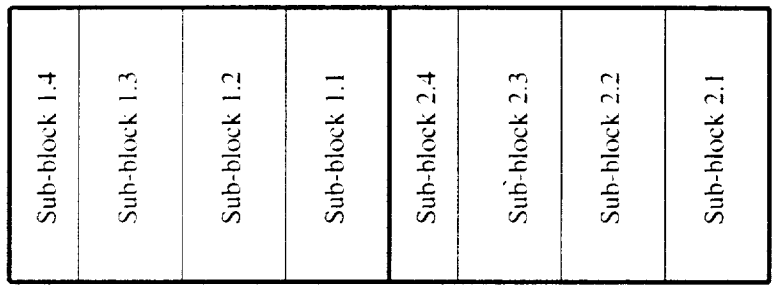


Figure III.1. Block division prior to merger.

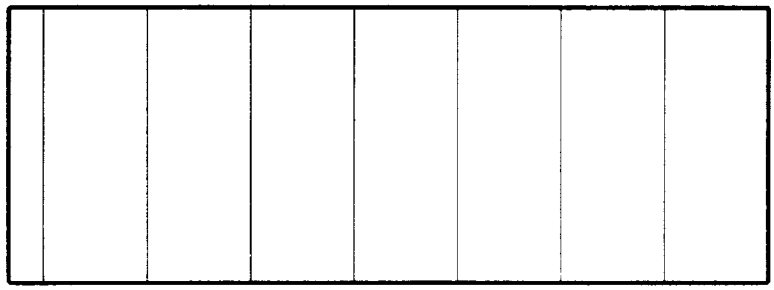


Figure III.2. Block after merger.

III.2. ALGORITHM

A very simple yet efficient algorithm based purely on the geometry of the multiblock grid and block connectivity was adopted for this program.

The input file containing the grid gives information only about the coordinates of the grid points, block dimensions component segments and boundary data. Hence the first task is to determine the block connectivity. This is done by analysing the false boundary information and determining blocks across opposite sides of the same false boundary segment. Once the interconnectivity between blocks is established, the total number of cells in the grid is calculated and that divided by the number of final subdivisions desired gives an estimate of the *average* number of cells per subdivision.

Based on this average value, blocks are classified into "small" blocks and "large" blocks. Large blocks are those whose size is greater than the average size determined above, whereas small blocks have a size less than or equal to the average size.

Large blocks are then split into smaller blocks, each of which has a size approximately equal to the average size. Smaller blocks are collected into groups so that the total size of each group is equal to the average size. This has been found to give excellent load balance for small grids, grids in which block sizes are compatible, and grids for unbladed configurations. For more complex grids satisfactory results have been obtained.

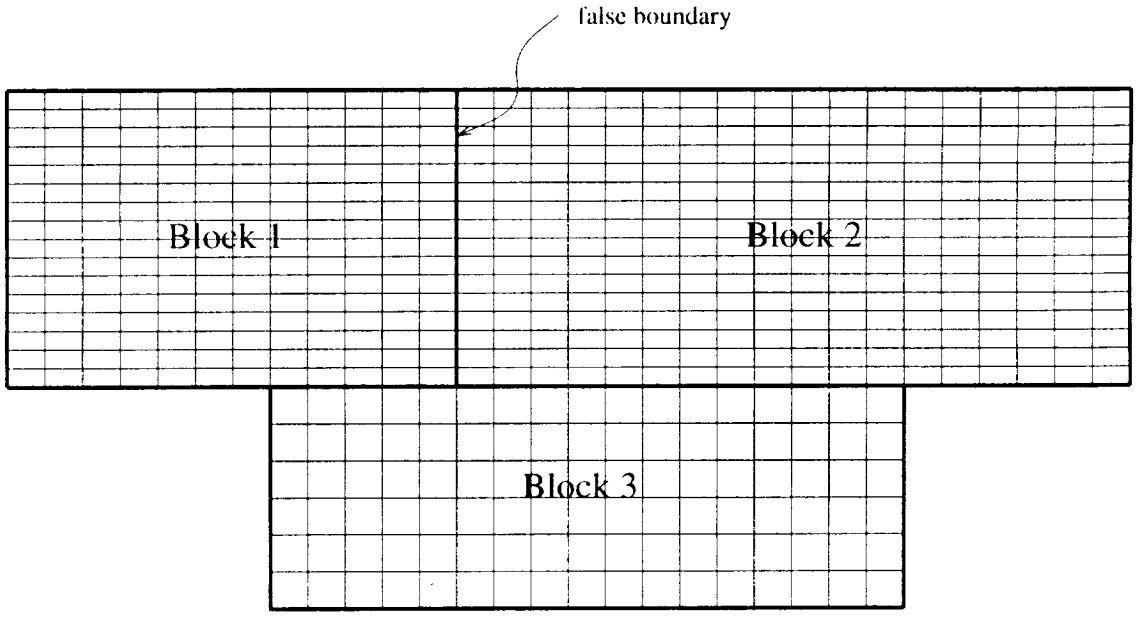


Figure III.3. Possible block mergers.

III.3 IMPLEMENTATION

III.3.1 Maximum Block Merger

As first step, blocks from the original grid are merged as much as possible to generated larger blocks. This is done so as to maximize processor usage.

Consider two blocks as illustrated in Figure III.1. Assume that load-balancing conditions require that each of the blocks be split into four blocks, vertically. Vertical splitting of blocks proceeds from right to left since blocks are stored in that manner. It is seen that sub-blocks 1.4 and 2.4 are clearly much smaller than the other sub-blocks and assigning an individual processor to each of them would be wasteful. Also, if the number of sub-blocks of each of the blocks is reduced to three from four, each of the processors will become overloaded. Therefore, the best solution to this problem is to first merge both the blocks and then split their combination to get a better load balance as shown in Figure III.2.

In this case it is seen that not only is the total number of sub-blocks reduced by one (implying that one less processor will be required) but also the load balancing is more effective since sub-block sizes are more compatible.

Blocks cannot be merged arbitrarily but only in those cases when the resulting larger block will have a regular grid. This is illustrated in Figure III.3. As can be seen in that figure, it is possible to merge block 1 with block 2 as their merging will result in the formation of a larger block in which the grid is regular. However, block 3 cannot be merged with either of block 1 or 2 as the resulting block would not have a top[ologically regular structure.

III.3.2. Block Classification

Once blocks are merged to the maximum permissible extent, all of them are passed through a classifying routine by which they are tagged as "small" or "large."

III.3.3. Splitting of "Large" Blocks

Those blocks classified as "large" blocks are split into smaller sub-blocks, each having a size as close as possible to the desired average size.

Blocks are split horizontally or vertically depending upon their dimensions. Wider blocks are split vertically and narrower blocks are split horizontally. Splitting of a block involves generation of a false boundary across the sub-blocks and checking for the presence of blades and other engine components which cannot be "cut". This is done to ensure continuity in these critical regions.

III.3.4 Re-merging and Grouping

Once all the "large" blocks are split, the second phase of the program begins in which the blocks are re-merged or grouped for maximum processor efficiency. This is done mainly to take very small blocks into consideration, which can be explained with the help of the Figure III.4.

In this situation, there is a cluster of "small" blocks with no "large" block nearby. The program selects a number of such "small" blocks and groups them into a cluster. Grouping is different from merging. Blocks are not merged to produce a single larger block but they are only meant to reside on the same processor. Grouping requires knowing block interconnectivity and proceeds in a recursive fashion. First, all "small" blocks adjacent to the present block are considered. Then, if the total number of cells of all the blocks is less than the average size, blocks adjacent to the block's adjacent blocks are examined. This goes on until a sufficient number of cells are obtained.

There is another case which arises after a "large" block has been split resulting in the generation of smaller sub-blocks, each being approximately equal to the average in size. In this case, again, the adjacent "small" blocks are merged into one of the children of the parent "large" block. Here, only one such block grouping is permitted since it should be noted that the "child" blocks are very close to the desired average size and the processor on which they reside should not be further loaded. This case is illustrated in Figure III.5.

III.3.5 Some Practical Considerations

It has been observed after going through several test cases that the process of load-balancing for complex multiblock grids is not always deterministic and hence user inputs may be required to make the process more predictable. This input comprises the following parameters.

MAX_TOL This is the maximum tolerable value to which a processor can be loaded, expressed in terms of the average size. Usually values between 0.95 and 1.2 give sufficiently good results.

MIN_TOL This is the minimum tolerable value to which a processor can be loaded, expressed in terms of the average size. Usually values between 0.6 and 0.9 give sufficiently good results.

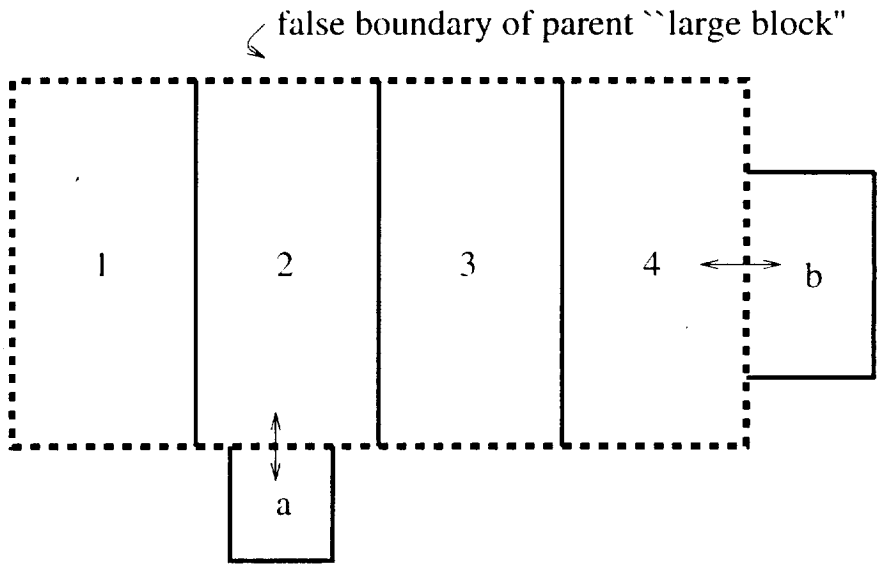
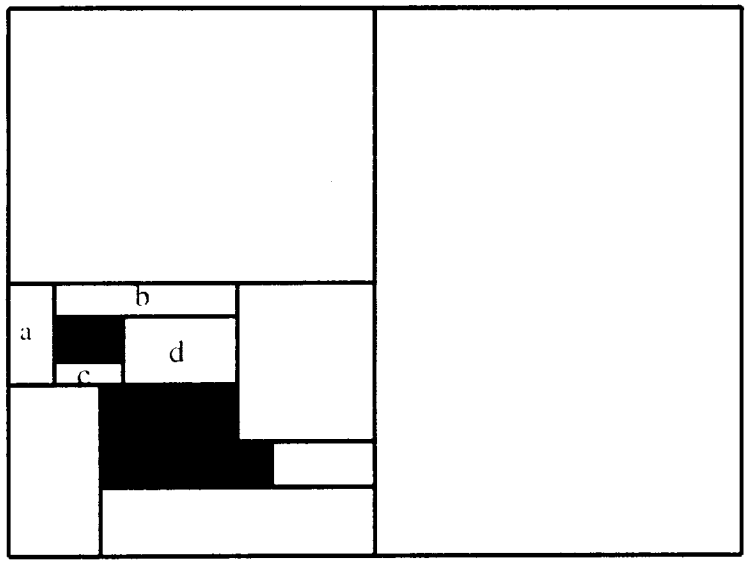


Figure III.4. Grouping of small blocks.



Blocks a, b, c and d would be ``clustered"

Figure III.5. Grouping of small and large blocks.

It should be noted that in most cases, the load balancing is independent of the above parameters. These attain significance only in case of very complicated geometries.

III.4 EXAMPLES

III.4.1 A Single Rectangular Block

This is the case of a single rectangular block having a regular grid. On this grid, a perfect load balance was obtained for an unbladed block, and satisfactory balance for a bladed block.

III.4.2 Two Adjacent Rectangular Blocks

The next test case considered two adjacent blocks, each of which contains a regular grids. Again, for this simple case, a perfect load balance was obtained for two unbladed blocks.

III.4.3 Grid for General Electric Energy Efficient Engine (GE-EEE)

This test case pertains to the Energy Efficient Engine model developed by General Electric. This model has been used extensively as computational-intensive tests for ENG10 [1]. The grid was generated using ENG10 preprocessors [2,3]. It contains 19 original blocks with approximately 115,000 grid points.

The initial load balance is only 15%, as the computational load is heavily dominated by the light-blue block of Figure III.6. This block contains a very fine grid because of the presence of a multistage compressor. A load balancing factor of approximately 80% was finally obtained. Stages of the load-balancing process carried out by LB for this fairly complex model are illustrated in color Figures III.6 through III.9.

III.5 CONCLUSIONS

A simple but effective algorithm for load-balancing discretizations consisting of multiple regular-grid blocks has been developed. Preliminary results suggest that the algorithm yields satisfactory results in the test cases considered here. These test cases have included a axisymmetric aerodynamic model of the complete GE-EEE, which has over 10^5 grid points. A load balance of approximately 80% was achieved for this demanding case.

A worthwhile refinement of LB would be the inclusion of *block weights* that account for computational intensity due to the presence of effects such as compression or combustion in specific regions. Such weights might be estimated from CPU measurements on sequential or vector machines, and fed to LB to further improve the decomposition logic.

Acknowledgement

Mr. Udayan Gumaste, who designed and wrote LB, would like to thank Dr. Mark Stewart for his support and assistance during the development of this program. He also acknowledges productive suggestions from Prof. Farhat. This work was supported by NASA Lewis Research Center under Grant NAG3-1425.

REFERENCES

1. M. L. Stewart, Axisymmetric aerodynamic numerical analysis of a turbofan engine, Draft paper, NASA Lewis Research Center, May 1994.

2. M. L. Stewart, Description of ENG10 subroutines and related programs, NASA Lewis Research Center, October 1994.

3. M. L. Stewart, A Multiblock Grid Generation Technique Applied to a Jet Engine Configuration, NASA CP-3143, NASA Lewis Research Center, 1992.

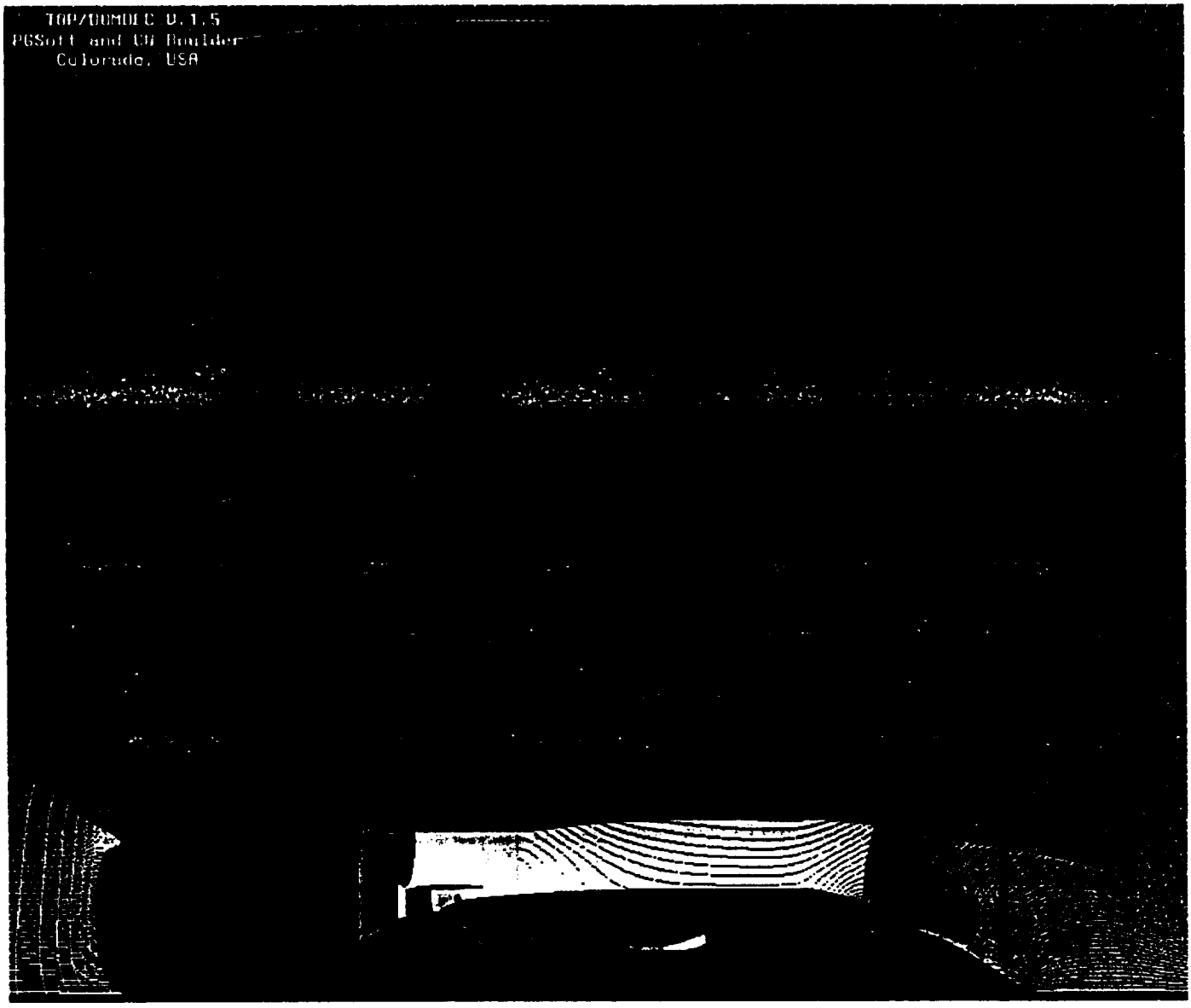


Figure II.6 : Initial Grid for GE–EEE ENG10 Model
(19 blocks, Load Balance : 0.1475)

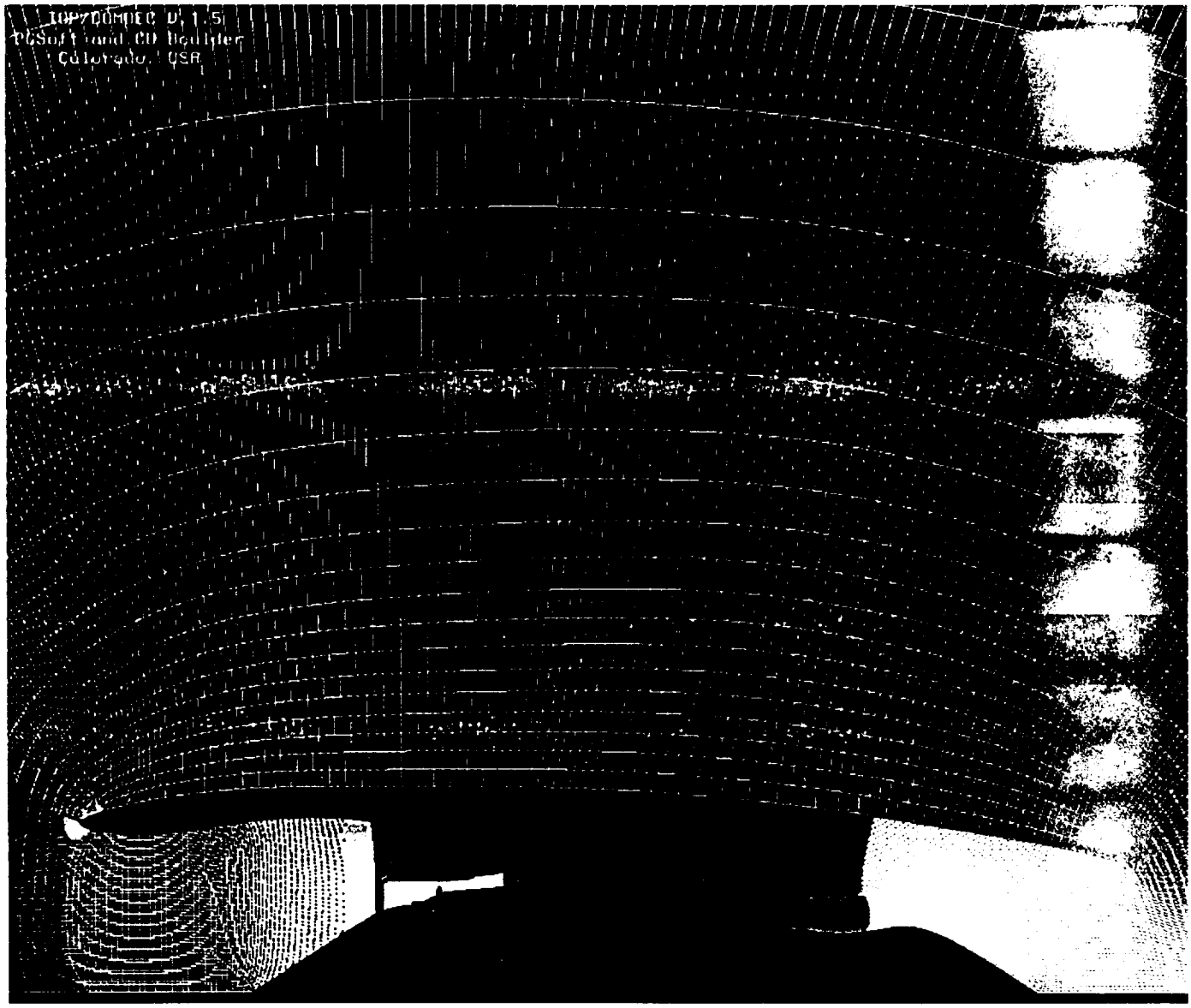


Figure II.7 : Grid for GE–EEE model after Maximum Merging of Adjacent Blocks
(Load Balance : 0.1123)

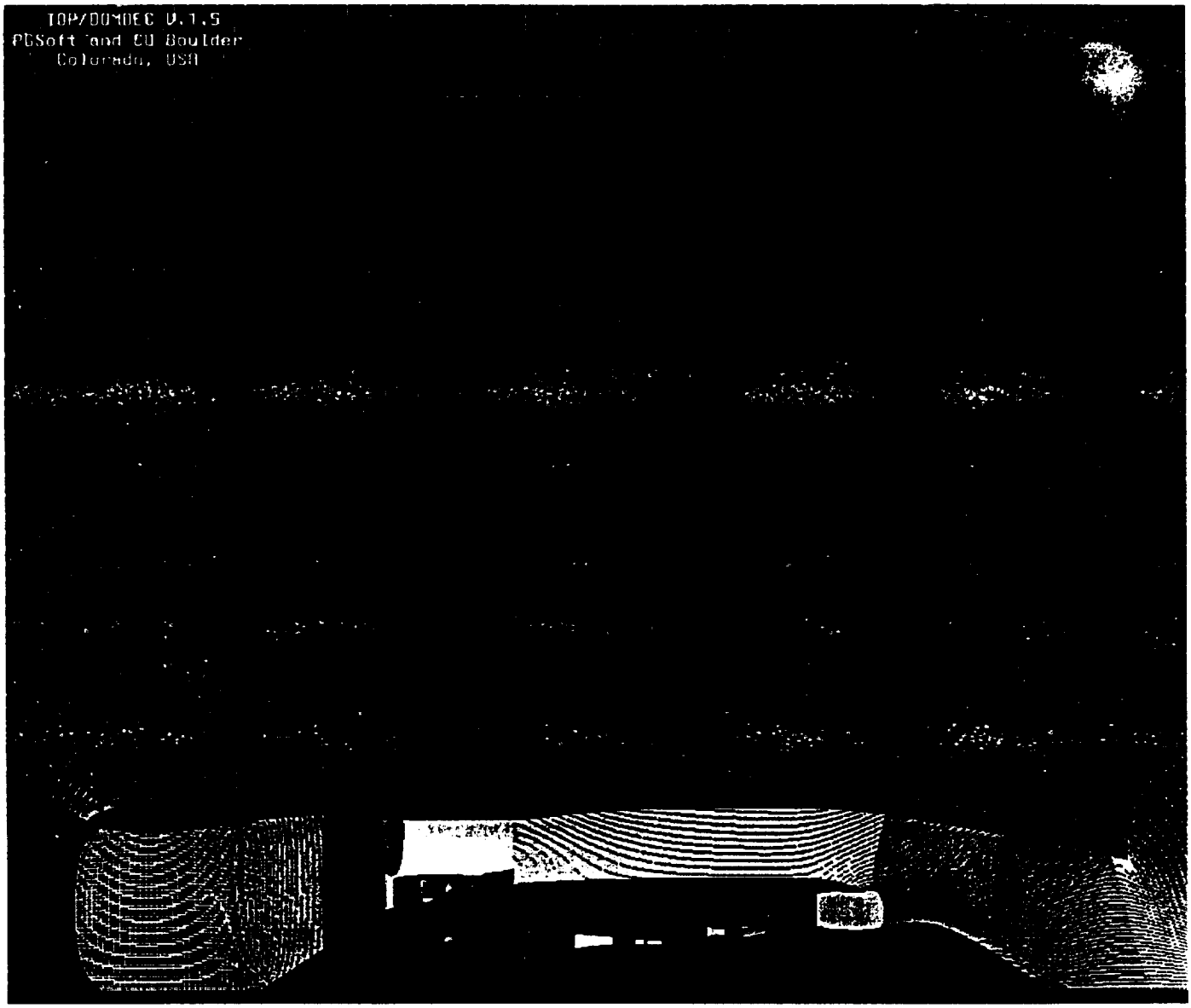


Figure II.8 : Grid for GE–EEE Model after Cutting Large Blocks
(Load Balance : 0.6553)

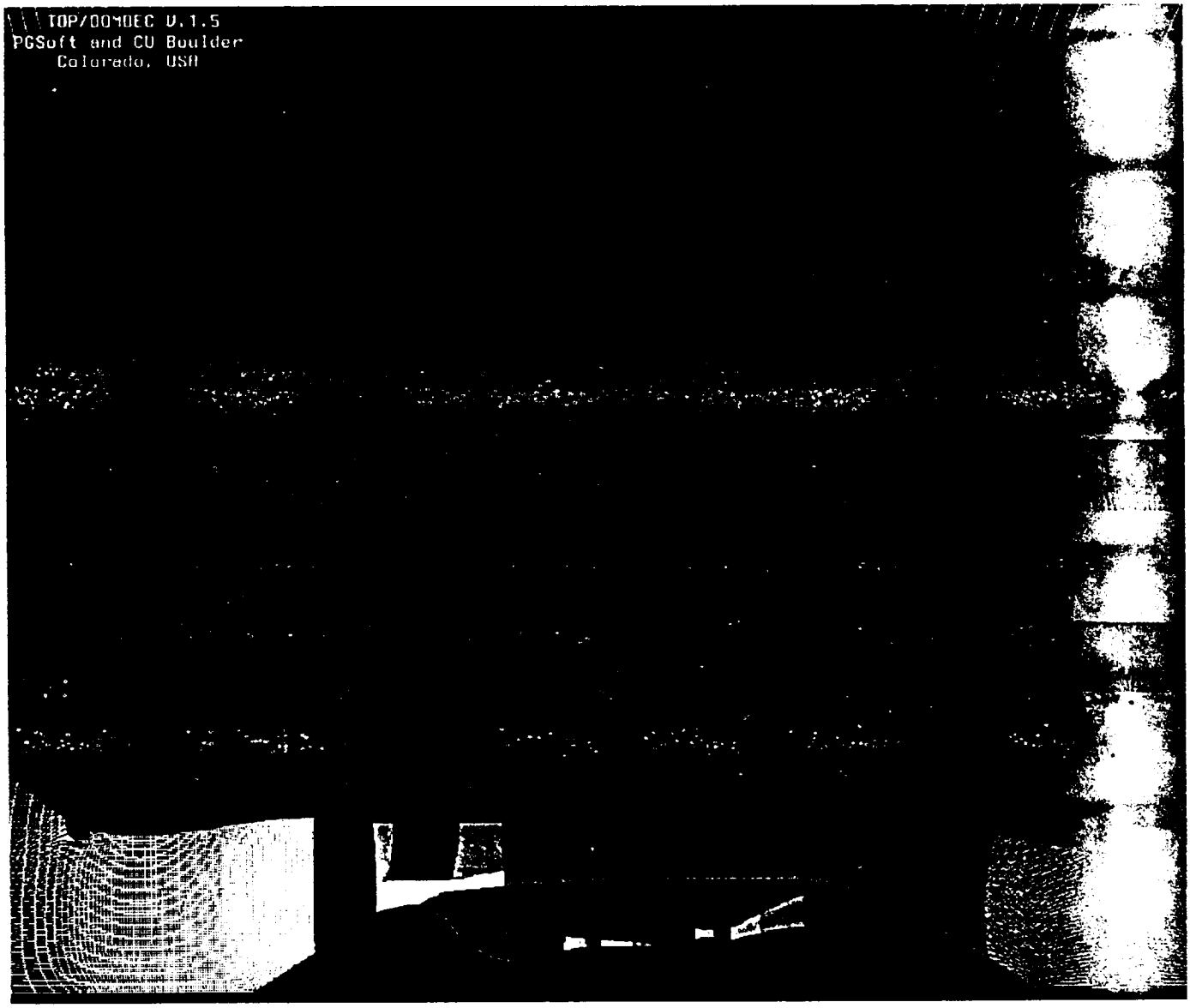


Figure II.9 : Final Configuration of GE–EEE Grid produced by LB
(Load Balance : 0.7943)

Appendix IV

Massively Parallel 3D Aeroelastic Analysis of Jet Engine

Udayan Gumaste, Carlos A. Felippa, and Charbel Farhat
Presented at the Computational Aerosciences Meeting
NASA Ames Research Center, Mountain View, CA, August 1996

This presentation reports progress in parallel computation methods for simulation of coupled problems applied to aircraft propulsion systems. This application involves interaction of structures with gas dynamics, heat conduction and heat transfer in aircraft engines. The methodology issues addressed include: discrete formulation of coupled problems; treatment of new effects due to interaction; staggered time stepping; scalable parallel solvers; and coarse three-dimensional versus fine two-dimensional models. The computer implementation issues addressed include: parallel treatment of coupled systems; domain decomposition and mesh partitioning strategies; mapping of decomposed models to hardware; and transfer of information between overall and regional models. The work is supported by NASA Lewis Research Center and monitored by Dr. C. C. Chamis.

A key objective is to demonstrate the application of this technology to achieve the first realistic unsteady aeroelastic analysis of a multirow-blade engine stage using three-dimensional models without making geometric approximations in advance. The first three-dimensional aeroelastic analysis involving a multiple fan-blade configuration was successfully performed during October 1995 on the NAS/IBM SP2 at NASA Ames. The aeroelastic model used for the simulation presented here comprises one half of a blade row that pertains to the compression stage of a GE EEE turbofan engine. This reduced but realistic configuration was used to test the fluid and structure mesh generators, mesh matchers and analysis modules. This test model has approximately 185,000 degrees of freedom. This simulation is a prelude to the treatment of more complex configurations involving two to four full-circle blade rows. Such models are expected to contain up to 2 million freedoms, which is close to the computational limit on present massively parallel computing platforms such as the IBM SP2 and Cray T3E.

The structure and fluid models for the test run are shown in the wireframes plots in Figures IV.1 and IV.2, respectively. The structure is treated by finite element shell model and processed by implicit integration with a FETI parallel solver. The fluid is treated by unstructured-mesh fluid volume methods stepped in time by MUSCL explicit solver. Structure and fluids advance with different time steps using a subcycle staggered solution scheme. As initial condition, a uniform longitudinal

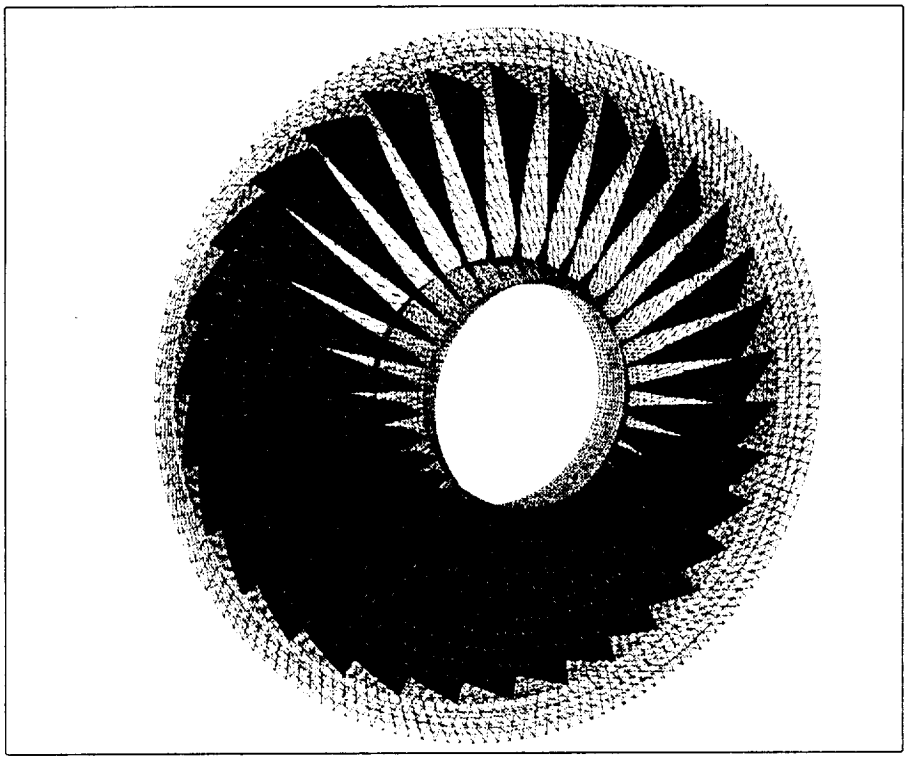


Figure IV.1. Structural model for the aeroelastic simulation of the 17-blade configuration.

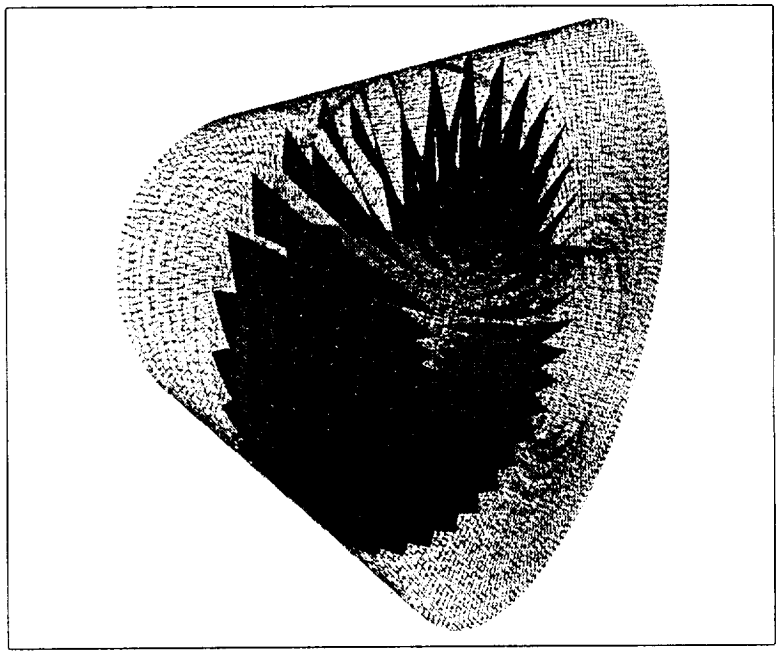


Figure IV.2. Fluid model for the aeroelastic simulation of the 17-blade configuration.

flow of 0.8M is applied to the nodes of the fluid mesh. It is left to runs through the rigid blades until a steady state is reached. Then the blades are released except for the end ones which are maintained fixed. The blades are set into motion by the transverse forces induced by their skew angles, were observe to vibrate approximately in phase. The total physical simulation time was 20 seconds, with 400 times steps performed in the structure and 8,000 steps on the fluid. Elapsed simulation time, using 28 SP2 processors was approximately 20 minutes. A color videotape of the dynamic response was prepared using the TOP/DOMDEC visualization system.

Appendix V

Massively Parallel Aeroelastic Analysis of Aircraft Engines

A Thesis Proposal for the Degree of Doctor of Philosophy

by

Udayan Anand Gumaste

Date : Mon., December 9, 1996

Center for Aerospace Structures and
Department of Aerospace Engineering Sciences
University of Colorado at Boulder
Boulder, CO 80309-0429

B.Tech., Indian Institute of Technology, Bombay, India, 1992
M.S., University of Colorado, 1995

Thesis Committee

Prof. Carlos A. Felippa	Aerospace Engineering Sciences/CAS	Chairman
Prof. Charbel Farhat	Aerospace Engineering Sciences/CAS	
Prof. K. C. Park	Aerospace Engineering Sciences/CAS	
Prof. C.-Y. Chow	Aerospace Engineering Sciences	
Prof. Oliver A. McBryan	Computer Science/CAPP	

Abstract

Aeroelasticity, or the interaction between the structure and the surrounding fluid is of particular importance in aerospace engineering. Catastrophic failures have been reported in which structures undergo severe damage by extracting energy from the flow and getting into unstable vibrations. This has been an active area of research for external flows for the past many decades and considerable progress has been made in the development of methods for analysis. However, turbomachinery aeroelasticity, though equally if not more important, has received attention since only recently on account of the higher degree of difficulty involved in analysis.

The aim of this research is to apply modern computational tools to simulate and analyze problems in turbomachinery aeroelasticity of aircraft engines. This is seen as a first step in the effort to perform a fully coupled multidisciplinary analysis of advanced propulsion systems. The central task of current research is to develop a methodology for simultaneous computer analysis of rotating and non-rotating components in turbomachines.

This proposal is divided into seven chapters and a bibliography.

Chapter 1 introduces the concept of aeroelasticity and enumerates different types of fluid-structure interactions. It also highlights differences between external and internal aeroelasticity.

Chapter 2 outlines the state-of-the-art in turbomachinery aeroelasticity with particular emphasis on the assumptions made to simplify analysis.

Chapter 3 begins with a brief explanation of the partitioned analysis approach to solving multidisciplinary coupled problems in engineering and describes a set of parallel three-dimensional computer programs currently available at the Center for Aerospace Structures to simulate such problems. A brief sketch of early attempts in using these programs "as-is" for turbomachinery simulations is also given.

Chapter 4 starts with the significance of analysis of rotor-stator interaction and points out the need to perform computations on discontinuous grids. It further gives a brief review of existing methods developed to enable fluid computations to be carried out on such grids.

Chapter 5 describes an unstructured two-dimensional fluid solver currently in use at the Center for Aerospace Structures with special emphasis on spatial discretization of the conservation equations. It then gives a detailed description on recent attempts to extend the capabilities of this solver to accept discontinuous grids and highlights features and draw-backs. A method to overcome current shortcomings is suggested at the end.

Chapter 6 consists of results using the extension of the fluid code mentioned in Chapter 5 for the shock tube problem. These are preliminary results covering different aspects of the newly developed methodology.

Chapter 7 concludes this proposal and outlines tasks for the future for completion of the current work towards a doctoral degree.

V.1 Introduction

Engines are the power plants of all aircraft systems. They are a marvel of man's advancement in science and technology in which the forces of earth, wind and fire are harnessed to propel gigantic flying machines through air in a small fraction of time as compared to other modes of transport. Designing engines is a daunting task, as the designer has to take into consideration the effects of changes in geometry and operational parameters on performance. Traditionally, design engineers have relied upon experimental methods and rig tests for design evaluation. This approach is not only time consuming but also very expensive as engines have to be rebuilt every time there is any change in their geometry. With the development of advanced computer algorithms and numerical methods and the availability of high-performance hardware, there is a growing trend towards the application of computing tools to engine design. Analysis of engine behavior to predict performance is challenging mainly on account of very complicated geometries, multiple moving components and most importantly the interactions between earth, wind and fire — the structure, the fluid flowing through the structure and thermal effects because of combustion. Amongst the three fields mentioned above, interaction between the structure and fluid components is of significance in the fan, compressor and turbine stages of an engine and is the focus of attention for current research.

V.1.1 Turbomachinery Aeroelasticity

Speed and economics of air transportation have been greatly benefited by the introduction of high-performance engines on aircraft systems. Increasing demands on performance have necessitated higher rotational speeds, thinner airfoils, higher pressure ratios per stage and increased operating temperatures. This has resulted in dynamic problems involving structural integrity, particularly those for bladed components of the engine. Such vibrations, induced by unsteady aerodynamic effects are generally classified as problems of turbomachinery *aeroelasticity*.

Aeroelastic vibrations in turbomachinery are usually of two types [4, 16], namely, forced vibrations such as those from upstream flow distortions or self-excited vibrations which are sustained by extraction of energy from the fluid.

V.1.1.1 Forced Vibrations

Forced vibrations in turbomachinery blading occur when blades are subjected to periodic aerodynamic forcing functions with frequency equal to a natural blade resonant frequency. One of the main sources for such forcing functions are upstream vanes, downstream vanes, distortion, rotating stall, surge, bleeds, mechanical sources and otherwise unidentified and random sources. The aerodynamic excitations are periodic mainly on account of spatially periodic variations in pressure, velocity and flow direction in the exit field of an upstream element which appear as temporally varying in a co-ordinate system fixed to the downstream blade row. As a result, individual blades are subject to a time-dependent forcing function which can induce high vibratory stresses.

V.1.1.2 Flutter

Under some conditions, a blade row operating in a completely uniform flow-field can get into a self-excited oscillation called *flutter*. The motion is sustained by the extraction of energy from the uniform flow during each vibratory cycle, with the flutter frequency corresponding generally to one

of the lower blade or coupled blade-disk natural frequencies. The outstanding feature of flutter is that very high stresses are generated within the blades leading to very short-term, high-cycle fatigue failures.

As problems of flutter and forced oscillation greatly affect engine performance, it is essential that these be predicted before engines are employed on aircraft systems. Traditionally, much of research conducted to this effect was based on empirical methods using engine and rig tests. These tests are time-consuming, expensive and at times risky as catastrophic engine failures are reported leading to accidents. With recent improvements in numerical algorithms and computer hardware, there is a growing trend towards using computational tools to analyze problems in turbomachinery aeroelasticity.

V.1.2 Turbomachinery vs. External Aeroelasticity

While considerable progress has been made in the computational analysis of aeroelastic phenomena for flows around external bodies, such as wings, wing-bodies or complete aircraft, that for aircraft engines and turbomachinery did not gather much momentum until the late 1970s and early 1980s [4]. One of the reasons for this delay was the complex nature of problems encountered in turbomachinery aeroelasticity which are summarized below [38] :

1. Large multiplicity of closely spaced mutually interfering blades, giving rise to both aerodynamic and structural coupling.
2. Presence of centrifugal loading terms both in the fluid and structural components.
3. Flow in blade cascades is much more complex than that in external flow cases on as it may be subsonic, sonic or supersonic depending upon the inlet Mach number and stagger angle giving rise to an intricate Mach reflection pattern.
4. Structural mistuning, which refers to slight differences in mode shapes or frequencies between the blades and can cause localized mode vibrations, in which all the energy in the system is concentrated on one or two blades leading to blade loss.
5. Aerodynamic mistuning, which refers to differences in blade-to-blade spacing and pitch angles altering the unsteady flow characteristics in blade passages.
6. For turbine blades, thermal effects will also have to be considered in addition to the interaction between fluid and structures.
7. The treatment of boundary conditions for fluid solvers is more complicated for internal flows than for external flows.
8. On account of moving components, structural analysis has to have geometric non-linearity capability.

V.2 Review of Existing Methods for Aeroelastic Analysis of Aircraft Engines

This section gives an overview of computational methods used for aeroelastic analysis of turbomachinery. Stress is laid on the assumptions made to simplify the analysis and make it tractable for computational methods. A brief summary of highlights of state-of-the-art methods will be given at the end.

V.2.1 Fluid Solvers

Early computer applications for turbomachinery problems focussed primarily just on predicting the flow pattern inside the system. This too presented major obstacles which could not be surmounted mainly on account of the lack of computing power at that time, circa early to mid 1970s. Researchers therefore resorted to making simplifying assumptions regarding fluid behavior in order to make these problems more amenable to computer solutions. These assumptions can be broadly classified into the following types :

1. Those made with respect to the 3-dimensional nature of flow.
2. Assumptions made to reduce the total problem size.
3. Assumptions made in mathematical modeling of fluid, i.e. the governing fluid equations.
4. Steady-state assumptions.

Each of these assumptions are further clarified below :

V.2.1.1 Assumptions Made with respect to the 3-Dimensional Nature of Flow

It should be noted that flow through turbine, compressor and fan rotors is inherently unsteady and 3-dimensional in nature. For example, large fan rotors have a velocity gradient from the hub, where the flow is subsonic, to the tip, where flow is supersonic as a result of blade rotation [5]. This, in addition to the variation of Coriolis forces in the radial direction gives rise to a very complex shock structure from hub to tip. Thus, in order to capture the true nature of flow, a fully 3-dimensional model is required.

However, on account of limitations in computing power, early researchers used simplified two-dimensional cascade models for flow computations. These models yielded sufficiently good results, in fact, to quote Bendiksen [4], "... it is surprising that [2-dimensional] cascade theories have been successful in 'explaining' — if not exactly predicting — the occurrence of flutter in supersonic fans ..."

While some purely 2-dimensional computations were carried out, more advanced flow solvers were developed based on a theory proposed by Wu [49] in 1952. In Wu's model, the flow is assumed to follow an axisymmetric streamsurface. The radius and thickness of this streamsurface are assumed to be known as a function of streamwise distance. These quantities are usually obtained from an axisymmetric throughflow or meridional analysis. The equations governing the flow along the streamsurface combine the axial and radial components into one streamwise component and are

thus 2-dimensional. The true 3-dimensional characteristics of flow can be extracted from this 2-dimensional approximation as the shape of the streamsurface is known. Specification of the streamsurface allows modeling of blades with variable heights and thicknesses, unlike that for the purely 2-dimensional solvers which had problems modeling blades of arbitrary shapes. As this approach uses 2-dimensional analysis to capture 3-dimensional phenomena, it is called "quasi 3-dimensional" and is common to many turbomachinery analysis programs.

V.2.1.2 Assumptions Made to Reduce the Total Problem Size

This assumption is common to many aeroelastic and fluid solvers of all types. For non-aeroelastic fluid solvers, it is obvious that flow through all interblade passages will be identical on account of similarity in geometry. Based on an interesting proposition of Lane [23] in 1957, even aeroelastic analyses, in which there is a change in geometry for each blade passage, can also be performed considering only one or a few interblade passages. This is highly beneficial as the total problem size is reduced by an order of magnitude.

Lane observed that at flutter, adjacent blades vibrate approximately 180 degrees out of phase with respect to each other. He considered the possible flutter mode shapes of a perfect rotor with identical blades and showed that the flutter mode shapes are remarkably simple : each blade vibrates with identical modal amplitudes but with a constant phase angle σ between adjacent blades. For a rotor with N blades, the possible interblade phase angles are given by :

$$\sigma_n = 2\pi n/N, \quad n = 0, 1, 2, \ldots, N-1$$

Thus the flutter mode is a traveling wave with respect to the rotor. This simple structure of the flutter mode is a direct consequence of the periodicity of cyclic symmetry in geometry which leads to important cyclic properties for both the structure and fluid. From a computational standpoint, Lane's Theorem, which assumes linear structural behavior, allows a full blade row of N blades to be modeled using only a single blade or a few blades.

V.2.1.3 Simplified Flow Models

For aeroelastic analysis there is a general consensus that viscous effects can be neglected except in stall and choke flutter [4]. Thus a 3-dimensional Euler solver would suffice. However, there is no general agreement on the ability of various formulations to capture the important features and stability characteristics of a given problem. Again, some assumptions are made to simplify the solution. These include :

1. Linearized Potential Flow : Two different classes of linearized unsteady cascade theories have been developed :

(a) Theories that linearize about a uniform mean flow.

(b) Theories that linearize about a non-uniform, deflected mean flow.

Of course, all these theories make the fundamental assumption that the flow is inviscid and of a perfect gas with no shocks. Bendiksen [4] has reviewed a large number of such flow solvers and these will not be repeated here.

2. Non-linear Flow Models : Some calculations of flow around cascades with non-linear potential models were reported in the early 1970s. However, these were rare and met with limited success.

Nowadays, with great advances being made both in the development of numerical algorithms and availability of powerful computing platforms, both Euler and Navier-Stokes solvers have become quite common and have been reported in significant numbers, for example [6, 7, 9, 17, 22] to name a few.

V.2.1.4 Time-Accuracy Assumptions

This is an assumption only when a fluid solver is used for aeroelastic analysis. Aeroelasticity is a truly unsteady phenomenon, yet at times, some researchers employed steady-state flow solvers for aeroelastic analysis. This is done by obtaining steady-state solutions from a flow solver and using that to perform a 'static' aeroelastic analysis.

V.2.1.5 Development of Advanced Flow Solvers

This is to give a very brief overview of the state-of-the-art in CFD for turbomachinery applications.

Keeping in phase with the development of CFD tools for external flows, commendable progress has been made in the development of advanced flow solvers for turbomachinery. Particular emphasis has been laid to develop sophisticated analysis methods to deal with the complex geometries of aircraft engines and difficulties arising out of that and the modeling of effects that other disciplines have on fluid flow.

In order to obtain fast steady-state flow solutions through complex aircraft engine geometries, advanced solvers are developed to reduce the large diversity between the length and time scales of flow. Prominent amongst these is the work of Adamczyk who uses advanced averaged models to compute flow in multistage turbomachinery. Three averaging operators are developed. The first averaging operator, namely the ensemble average, is introduced to eliminate the need to resolve the detailed turbulent structure of flow. The second operator is used for time-averaging and allows fast computation of steady flows. The last operator, namely the passage-to-passage averaging operator allows simultaneous simulation of flows through blade-rows having variable number of blades and/or rotating speeds. Details of these operators are lengthy and complex and will not be dealt with in this report. The reader is referred to [1] for the full mathematical formulation.

With growing interest in treating aircraft engines on a more global basis, particular emphasis is laid on modeling interdisciplinary interaction between fluid and other components. Stewart [44] has developed a program `ENG10` which takes into account the effect of blade forces, loss, combustor, heat addition, blockage, bleeds and convective mixing. This program, in the writer's opinion, represents the true state-of-the-art in turbomachinery flow solvers and can be viewed as an efficient synthesis of existing models for multidisciplinary interaction. An approach similar to that of Adamczyk is used, in which the right-hand sides of Euler equations include averaged terms for blade forces, combustor and other effects mentioned above.

Other notable works in this area are those of Koya and Kotake [22] and Gerolymos [17, 18]. Of these Koya and Kotake are credited the first truly 3-dimensional time-dependent Euler calculation for flow through a turbine stage. Gerolymos developed advanced methods for investigation of flutter in vibrating cascades, employing assumptions made about linear structure behavior and spatial periodicity.

V.2.2 Structure Solvers

The development of structure solvers for aircraft engine applications has not been much different from that for any other structural analyses. In fact, Reddy *et al* [38] mention that most of the structural calculations at NASA LeRC have been performed using NASTRAN.

Some specific stand-alone programs, especially those which take into account thermal and other effects such as bird and ice impacts and also the effects of composites used have also been used for blade analysis though not directly coupled with a fluid solver for aeroelastic analyses [37].

V.2.3 Summary of Aeroelastic Analysis Programs

Some of the assumptions made for aeroelastic analysis of turbomachinery have been mentioned above. The following is a brief summary of research in turbomachinery aeroelasticity till now [2, 38, 42] :

1. Use of potential or Euler solvers with simplifying assumptions .
2. Purely 2-dimensional, quasi 3-dimensional or axisymmetric fluid solvers.
3. Only one or a few blades are modeled.
4. To compute structural response, linear structural behavior is assumed. This makes it possible to use quicker frequency domain analysis.
5. Very often, it is found that only static structure response is considered, neglecting inertia effects, even when the method of analysis is for unsteady analysis.
6. For cases in which inertia effects are considered, a very simplified structural model is used, with as few as 2 DOFs.
7. Even though aeroelasticity is an unsteady phenomenon, steady-state methods are used to compute fluid flow and structure loads are computed at each time step from this steady state solution. As time-accuracy of fluid solvers is sacrificed in order to obtain a fast steady-state solution, this may not yield correct results.
8. Transfer of loads between fluid and structure is done through lift coefficients, thus losing spatial accuracy in computing structural loads.
9. Some fluid solvers use moving meshes for analyzing vibrating blades. Exact details of algorithms for mesh updating are not given and it is probable that these algorithms do not satisfy the geometric conservation law, which will be discussed later.

V.3 Partitioned Analysis Procedures for the Aeroelastic Problem

This chapter deals with the formulation of coupled field problems for aeroelasticity and their solution using the partitioned analysis approach. It begins by introducing the concept of partitioned analysis and the motivation behind this methodology. Use of partitioned analysis for aeroelastic applications will be mentioned and elaborated upon. The individual software components used for solving the coupled field aeroelastic problem will be briefly overviewed. Lastly, a brief description of initial attempts at using existent technology for external aeroelasticity for internal aeroelasticity applications will be given.

V.3.1 Partitioned Analysis and Coupled Field Problems

Many current problems in engineering require the integrated treatment of multiple interacting fields. These include, for example, fluid-structure interaction for submerged structures and in pressure vessels and piping, soil-water-structure interaction in geotechnical and earthquake engineering, thermal-structure-electromagnetic interaction in semi- and superconductors and fluid-structure interaction (FSI) in aerospace structures and turbomachinery, the last of which is the focus of attention for current research.

Nowadays, sophisticated and advanced analysis tools are available for individual field analysis. For example, for FSI, advances in the last few decades have resulted in the development of powerful and efficient flow analyzers, which is the realm of interest of computational fluid dynamics (CFD). Equally robust structural analysis tools are available, which is a result of development of advanced finite element methods (FEM). Computer analysis of coupled field problems is a relative newcomer and no standard analysis methodology has been established. One natural alternative is to tailor an existing single-field analysis program to take into account multidisciplinary effects. As an example, fluid volume elements could be added to a FEM structure solver. Another approach would be to unify the interacting fields at the level of governing equations and formulate analysis methods thereupon, for example, as suggested by Sutjajho and Chamis [45].

Both these methods suffer from drawbacks. From a programming point-of-view, addition of modules of different fields leads to an uncontrolled growth in complexity of necessary software. It becomes increasingly difficult to modify existing codes to incorporate improved formulations. For users, a monolithic code can impose unnecessary restrictions in modeling and grid generation. For example, in FSI, forcing equal grid refinement on the fluid-structure interface may either cause the structure elements to be too small, making analysis more expensive, or cause fluid cells to be too large, resulting in a loss of accuracy and/or stability.

Partitioned analysis [31] offers an attractive approach in which diverse interacting fields are processed by separate field analyzers. The solution of the coupled system is obtained by executing these field analyzers, either in sequential or parallel manner, periodically exchanging information at synchronization times. This approach retains modularity of software and simplifies development. It also allows to exploit well established discretization and solution methods in each discipline and does not enforce any specific grid refinement requirements.

V.3.2 Partitioned Analysis for Aeroelastic Applications

Aeroelasticity deals with the interaction of high-speed flows with flexible structures. Thus, in a physical sense, it is a two-field phenomenon. However, on account of different formulation methods used for the fluid and structure components, computationally, it becomes more convenient to treat this as a three-field coupled problem.

V.3.2.1 Aeroelasticity as a Three-Field Coupled Problem

Traditionally, structural equations are formulated in Lagrangian co-ordinates, in which the mesh is embedded in the material and moves with it; while the fluid equations are written in Eulerian co-ordinates, in which the mesh is treated as a fixed reference through which the fluid moves.

Therefore, in order to apply the partitioned analysis approach, in which the fluid and the structure components are treated separately, it becomes essential to move at each time step, at least the portions of the fluid grid that are close to the moving structure. One of the approaches which obviates the need for partial regridding of the fluid mesh is one where the moving fluid mesh is modeled as a pseudo-structural system with its own dynamics. Thus, the *physical* two-field aeroelastic problem can be *computationally* formulated as three-field system, comprising of the fluid, the structure and the dynamic mesh. This is the Adaptive Lagrangian-Eulerian (ALE) [8, 27] formulation. The semi-discrete equations governing this three-way coupled problem can be written as follows :

$$\frac{\partial}{\partial t}\left(\mathbf{A}(\mathbf{x}, t)\mathbf{w}(t)\right) + \mathbf{F}^{c}\left(\mathbf{w}(t), \mathbf{x}, \dot{\mathbf{x}}\right) = \mathbf{R}\left(\mathbf{w}(t)\right) \tag{3.1a}$$

$$\mathbf{M}\frac{\partial^2 \mathbf{q}}{\partial t^2} + \mathbf{f}^{int}(\mathbf{q}) = \mathbf{f}^{ext}\left(\mathbf{w}(t), \mathbf{x}\right) \tag{3.1b}$$

$$\tilde{\mathbf{M}}\frac{\partial^2 \mathbf{x}}{\partial t^2} + \tilde{\mathbf{D}}\frac{\partial \mathbf{x}}{\partial t} + \tilde{\mathbf{K}}\mathbf{x} = \mathbf{K}_c\mathbf{q} \tag{3.1c}$$

where t designates time, $\mathbf{x}$ is the position of a moving fluid grid point, $\mathbf{w}$ is the fluid state vector, $\mathbf{A}$ results from the finite-element/finite-volume discretization of the fluid equations, $\mathbf{F}^c$ is the vector of convective ALE fluxes, $\mathbf{R}$ is the vector of diffusive fluxes, $\mathbf{q}$ is the structural displacement vector, $\mathbf{f}^{int}$ denotes the vector of internal forces in the structure, $\mathbf{f}^{ext}$ the vector of external forces, $\mathbf{M}$ is the finite element mass matrix of the structure, $\tilde{\mathbf{M}}$, $\tilde{\mathbf{D}}$ and $\tilde{\mathbf{K}}$ are fictitious mass, damping and stiffness matrices associated with the moving fluid grid and $\mathbf{K}_c$ is a transfer matrix that describes the action of the motion of the structural side of the fluid-structure interface on the dynamic fluid mesh. For example, $\tilde{\mathbf{M}} = \tilde{\mathbf{D}} = 0$ and $\tilde{\mathbf{K}} = \tilde{\mathbf{K}}^R$ where $\tilde{\mathbf{K}}^R$ is a rotation matrix corresponds to a rigid mesh motion of the fluid grid around an oscillating structure, while $\tilde{\mathbf{M}} = \tilde{\mathbf{D}} = 0$ includes the spring-based mesh updating scheme proposed by Batina [3] and Tezduyar *et al*[46].

It should be noted that the three components of the coupled field system described in (3.1) exhibit different mathematical and numerical properties and hence require different computational treatments. For Euler and Navier-Stokes flows, the fluid equations are non-linear. The structural equations may be either linear or non-linear depending upon the type of application. The fluid and structure interact only at their interface, via the pressure and motion of the structural interface. However, the pressure variable cannot be easily isolated from the fluid equations or the fluid state vector $\mathbf{w}$, making the coupling in this three-field problem implicit rather than explicit.

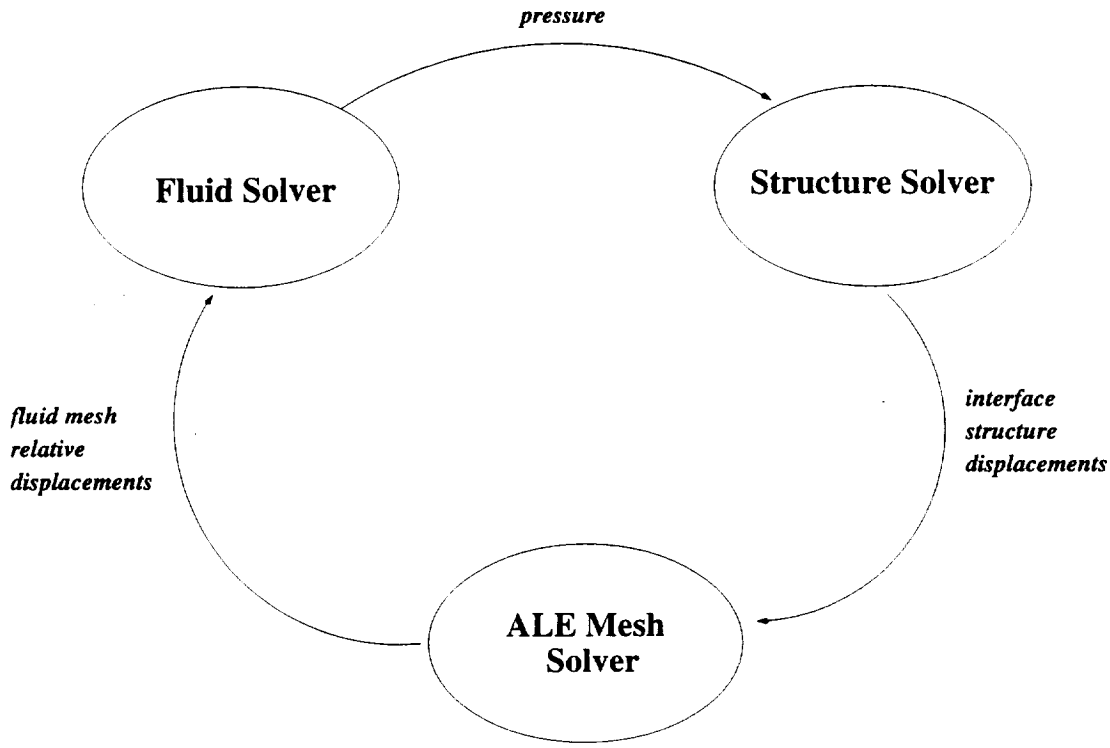


Figure V.3.1 : Interaction between Programs for FSI

The simplest possible partitioned analysis procedure for transient aeroelastic analysis is as follows :

(a) Advance the structural system under a given pressure load.

(b) Update the fluid mesh according to the movement of the fluid structure interface.

(c) Advance the fluid system and compute the new pressure load.

This procedure is carried out in cyclic order until the desired end of computations is reached, see Figure V.3.1.

V.3.2.1.1 Geometric Conservation Law

An interesting feature that arises out of the use of the three-field ALE formulation is the need to take into consideration the motion of fluid volume cells while computing fluxes in the fluid solver. It is shown in [47] that in order to compute flows correctly on a dynamic mesh, it is essential that the selected algorithm preserves the trivial solution of an uniform flow-field even when the underlying mesh is undergoing arbitrary motions. The necessary condition for the flow solver to accomplish this is referred to in literature as the Geometric Conservation Law (GCL). Failure to satisfy the GCL results in spurious oscillations although the system for which solution is sought is physically stable.

V.3.3 The PARFSI System for Unsteady Aeroelastic Computations

A system of locally developed programs for unsteady aeroelastic computations, PARFSI (**Par**allel

Fluid-Structure Interaction) will be described next. This system consists of a fluid solver, a structure solver, a dynamic ALE mesh solver and a few preprocessing programs for parallel computations.

V.3.3.1 Fluid Solver

For flow computations, a 3-dimensional fluid solver for unstructured dynamic meshes is used. This discretizes the conservative form of the Navier-Stokes equations using a mixed finite-element/finite-volume (FE/FV) method. Convective fluxes are computed using Roe's [39] upwind scheme and a Galerkin centered approximation is used for viscous fluxes. Higher order accuracy is achieved through the use of a piecewise linear interpolation method that follows the principle of MUSCL (Monotonic Upwind Scheme for Scalar Conservation Law) proposed by Van Leer [26]. Time integration can be performed either explicitly using a 3-step variant of the Runge-Kutta method, or implicitly, using a linearized implicit formulation. An elaborate description of the 3-dimensional fluid solver can be found in [24].

V.3.3.2 Structure Solver

A parallel structural analysis program, PARFEM has been developed by Farhat and co-workers over the last few years. This program has a wide range of one-dimensional to three-dimensional finite elements for structural analysis. Time-integration is implicit based on Newmark's method. For parallelization, the FETI (Finite Element Tearing and Interconnecting) [10, 11] domain-decomposition method is used.

V.3.3.3 ALE Mesh Solver

The fluid mesh is assumed to be a network of springs based on a method proposed by Batina [3]. The solver used to update the fluid mesh is integrated into the fluid code as a subroutine which is called every time there is an exchange of information between the structure and fluid. At each time step t^{n+1}, displacements at the interior grid points are predicted by extrapolating the previous displacements at time steps t^n and t^{n-1}. Grid points on the far-field boundaries are held fixed, while the motion of grid points on the fluid-structure interface is obtained by interpolation of structural displacements.

V.3.3.4 Preprocessing Programs

Two preprocessing programs have been developed to enable parallel aeroelastic computations. To decompose the fluid and structure meshes, a mesh decomposition software TOP-DOMDEC [13] is used. This is equipped with a range of mesh decomposition algorithms and can also be used as a visualization tool.

As fluid and structure computations are performed by independent programs adhering to the partitioned analysis methodology, the fluid and structure meshes need not coincide along their interfaces. Hence an interpolation procedure is followed to transfer pressures from the fluid to the structure and displacements from the structure to the fluid. Interpolation information (in terms of interpolation coefficients within elements and association of fluid/structure nodes/elements across the fluid structure interface) necessary for parallel execution of solvers is set up by a preprocessing program MATCHER, described in [30].

V.3.3.5 Subcycling between Fluid and Structure Solvers

The fluid and structure meshes may have varied degrees of refinement and will hence have different

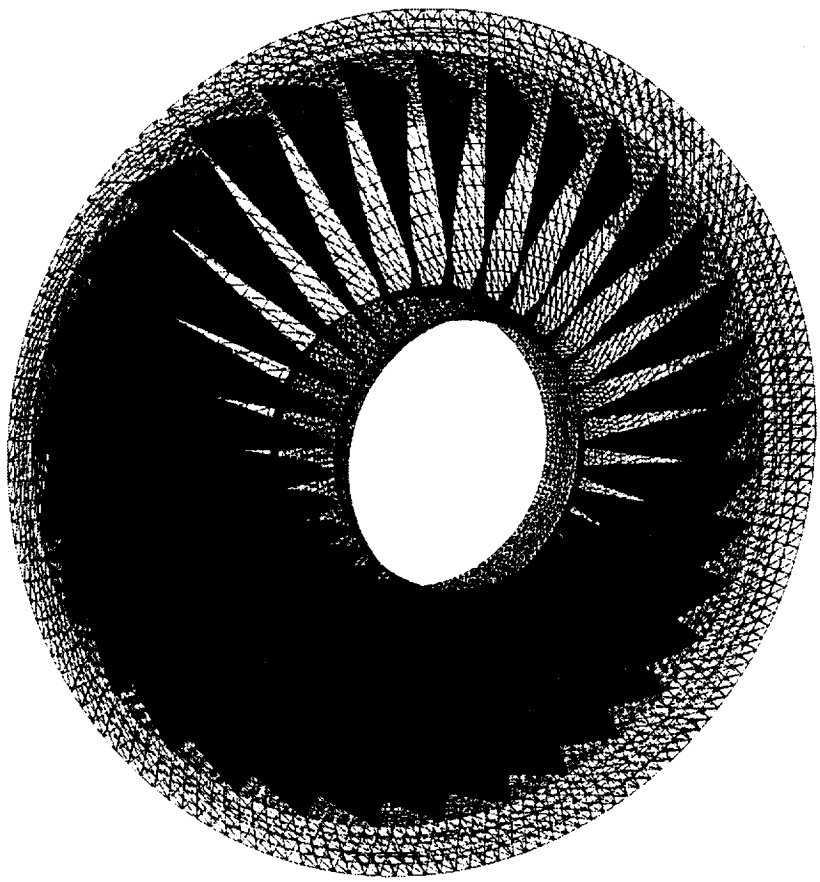


Figure V.3.2 : Fluid & Structure Meshes for the GE-EEE Fan Stage

time steps. Subcycling [32] allows the fluid and structure solvers to run concurrently with different time steps by periodic exchange of information at synchronization times. This also makes structural computations more efficient as usually the implicit structure time step is an order of magnitude higher than the explicit fluid time step.

V.3.4 Application of PARFSI for Turbomachinery Simulations

As a beginning, the existing programs for aeroelastic analysis were used to simulate the aeroelastic response for the blades of the GE-EEE fan stage [20]. Disregarding modifications made to some pre- and post-processing programs, no major modifications were required for any of the field analyzers in computing the response to internal flow using codes primarily designed for external aeroelastic computations. This highlights a major benefit of adopting the partitioned analysis methodology.

Two physical models have been used in the aeroelastic simulations.

1. The first model is a single row of blades from the compression stage of the GE-EEE turbofan engine, which serves as a testcase for most computational methods at NASA LeRC. This model consists of 32 blades along the circumference. Details of blade geometry were obtained from a NASTRAN FE model provided by Scott Thorpe of NASA LeRC. This model has approximately 60,000 fluid nodes and 1,600 structure nodes. For parallel analysis, the fluid mesh was decomposed into 32 sub-domains and the structure mesh into 4 subdomains.
2. The second test model was a hypothetical two-row stage which was obtained by using the GE-EEE model mentioned above, setback along the longitudinal axis of the engine and half-way shifting it in the circumferential direction. In this case, the fluid mesh consisted of approximately 45,000 nodes and the structure mesh has approximately 3,200 nodes. For parallel analysis, 16 sub-domains were used for the fluid and 4 for the structure.

Meshes for both models were built by first constructing a mesh for a single cell block in which the blade profile was swept around the circumference to obtain hexahedra which were further divided into tetrahedra for the fluid volume. For the structure, blade profiles were rotated around the circumference and divided into triangular shell elements. Wireframe plots of the fluid and structure meshes generated for each of the above cases are shown in Figures V.3.2 and V.3.3.

V.3.4.1 Results

It was observed that the blades tend to vibrate in phase with similar amplitudes. A slight coupling effect was observed between the bending and torsional modes of vibration for the blades.

Results for the two-row case were more interesting. The first row appeared to act as a screen and absorbed most of the impact of the aerodynamic load. This caused it to vibrate with a much greater frequency and amplitude than the second row. Again, some bending-torsion coupling was observed in blade vibrations.

V.3.4.2 Shortcomings

Although `PARFSI` is designed mainly for external aeroelastic simulations, it can be tailored to perform internal aeroelasticity computations without having to make any significant modifications to the existing codes and/or methodologies.

To make it better applicable to turbomachinery problems, the following enhancements need to be made :

1. Addition of rotational source terms in fluid and structure solvers.
2. Addition of geometric non-linear analysis capability to take into account large rotational rigid-body displacements of blades.
3. Make the fluid solver cable of handling differential rotations between rotating and non-rotating components.
4. Addition of thermal effects.

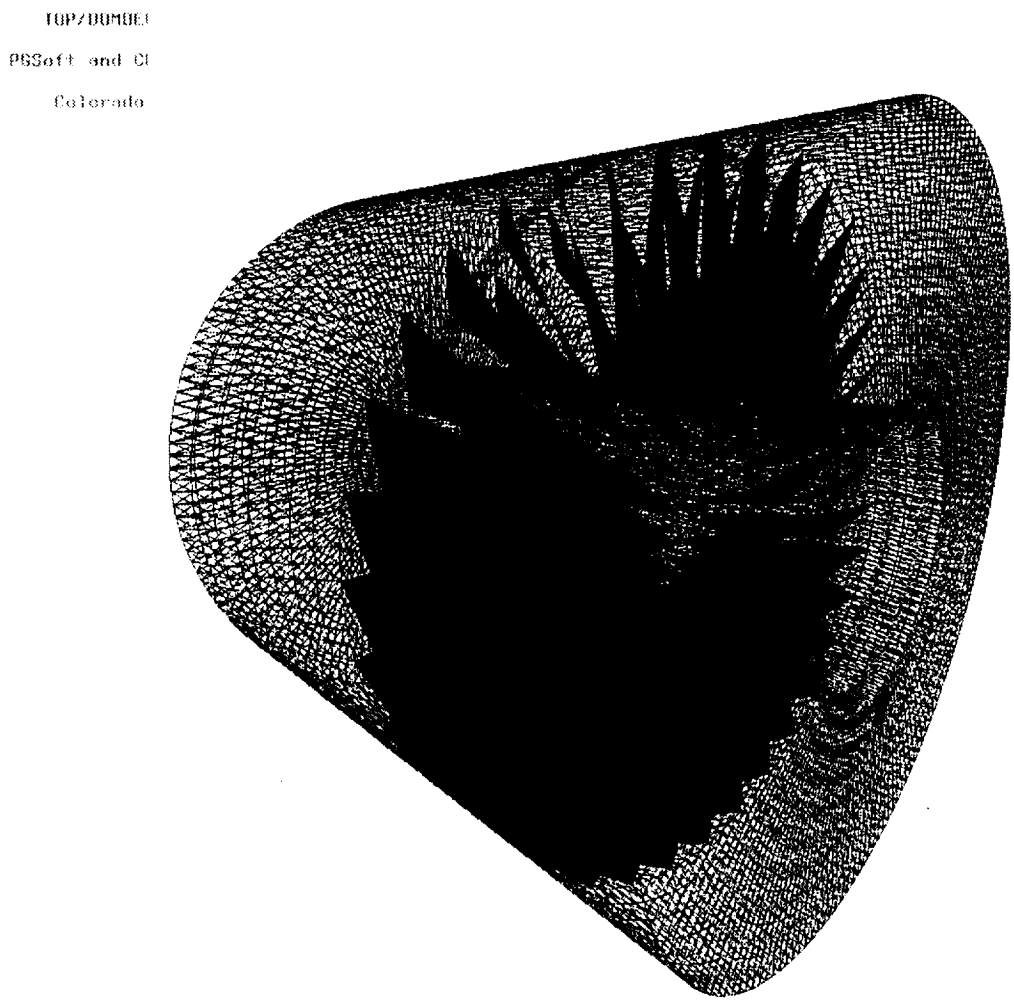


Figure V.3.3 : Fluid & Structure Meshes for the Hypothetical 2-Row Stage

V.4 Computational Analysis of Rotor-Stator Interaction

A *stage* in an aircraft engine is usually made up of a rotating component, the *rotor* and a non-rotating component, the *stator*. The function of the rotor is to add energy to the flow by mechanical interaction of the fluid with the blades. In this process the fluid acquires angular momentum. The stator removes this angular momentum and diffuses flow to raise pressure. This combined action of the rotor and the stator is of fundamental importance to the performance and efficiency of the engine and hence is a matter of key research interest.

An engine usually contains several such rotor-stator stages and hence the ability to analyze such stages forms the first building block in an attempt to simulate a whole aircraft engine. From an aeroelasticity or FSI point-of-view, mutual interactions between rotors and stators (rotor-stator interaction or RSI) become important when the axial gap between these two components is made smaller in order to reduce the overall engine length. Experimental results indicate that flows become unsteady on account of the interaction of the downstream airfoils with the wakes and passage vortices generated upstream, from the motion of rotors relative to the stators and from vortex shedding at blunt airfoil trailing edges. This unsteady interaction affects the aerodynamic, thermal and structural performance in each stage and hence ultimately the engine as a whole.

Another topic of interest would be interaction of flow and mechanical components between non-rotating components such as inlets and diffusers with a rotating component such as a large fan.

In all these cases, the area of investigation is the nature of flow as it undergoes transition from a non-rotating to a rotating flow regime and vice-versa. Following are the two main issues that must be addressed for successful computational analysis of such phenomena :

1. As the flow passes from a non-rotating to a rotating flow regime it will experience a sudden change in circumferential momentum. Care will have to be exercised in developing a numerical method for such cases so that it does not create any artificial numerical shocks.
2. For aeroelastic computations with dynamic meshes as mentioned in Section V.3.3, mesh lines will no longer be continuous between fluid meshes of rotating and non-rotating components, see Figures V.4.1 and V.4.2. Hence some interpolation method will have to be developed to handle such situations.

This section gives an overview of a few methods common for analyzing flows on discontinuous grids and highlights their merits and demerits.

V.4.1 Flow Computations on Discontinuous Grids

Grid generation and subsequent treatment of moving grids was an area of difficulty which early researchers faced in simulating rotor-stator flows. A single grid wrapping around both the rotor and the stator has to distort considerably to accommodate the motion of the rotor and could result in inaccurate calculations.

One suggested alternative to overcome this problem is the use of zonal grids in which the region of interest is divided into several geometrically simpler subregions (or zones). This makes both mesh

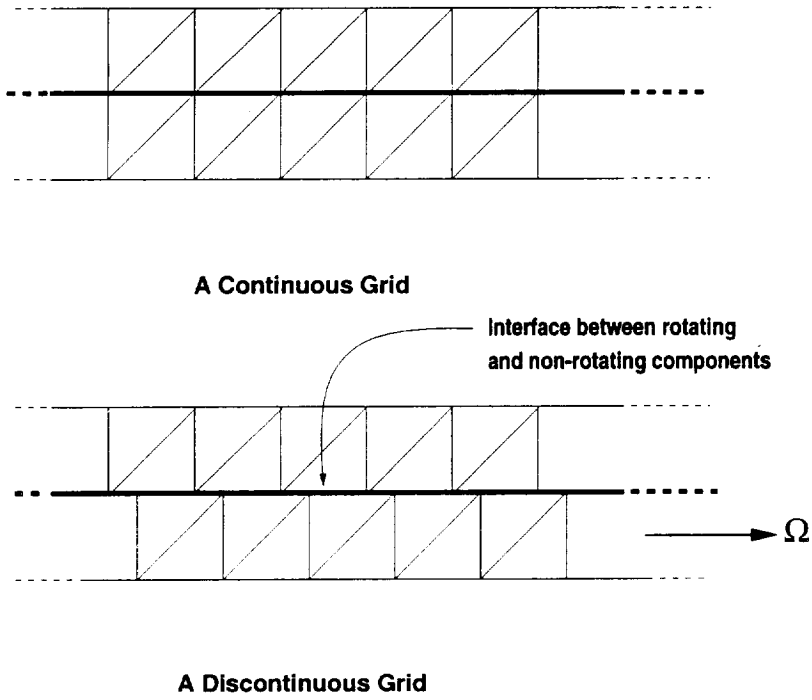


Figure V.4.1 : Grid Discontinuity Arising in RSI Computations

generation and treatment of large grids for complex geometries easier.

However, in case of simulations where there is relative motion between adjacent sub-grids, some method has to be developed in which there is a smooth and accurate exchange of information between grids whose lines are no longer continuous. Two such approaches exist. One is the used of overlaid grids in which two zones overlap each other and the exchange of information takes place in the region of overlap. Another approach is to use patched grids in which information transfer occurs at the interface between connected zones. Examples of overlaid and patched grids for an inner cylinder and outer square are seen in Figures V.4.3 and V.4.4.

While each of the patched and overlaid grid approaches has their own advantages and disadvantages, patched grids are preferred over overlaid grids for the following reasons :

1. Overlaid grids incur higher interpolation costs as a problem in n spatial dimensions requires an interpolation in n dimensions whereas that for patched grids would require interpolation only in $(n-1)$ dimensions as exchange of information takes place only at zonal interfaces.
2. Development of conservative zonal schemes for overlaid grids is more difficult than that for patched grids. This makes use of overlaid grids less suitable for computations which contain sharp discontinuities in flow.
3. The accuracy and convergence speed of the calculation seems to depend on the degree of overlap of the zones and the relative size of each zone, thus introducing a certain amount of undesirable empiricism in the formulation.

Numerical methods developed for treatment of zonal patched grids must satisfy several requirements

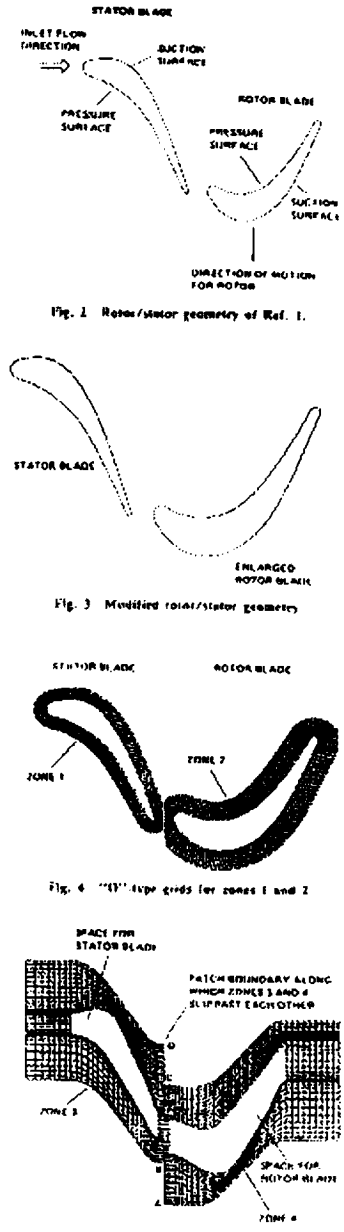


Figure V.4.2 : Grid Generation and Discontinuity for Rotor-Stator Interaction

before they can be used effectively :

1. numerical stability
2. spatial and temporal accuracy

Figure V.4.3 : Example of a Patched Grid for an Inner Cylinder and Outer Square

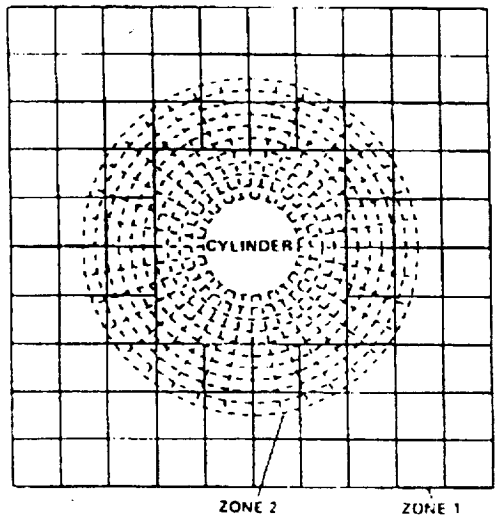


Figure V.4.4 : Example of a Overlaid Grid for an Inner Cylinder and Outer Square

3. developed scheme should be conservative so that flow discontinuities can move from one grid to another without distortion

In the past few years, several zonal boundary schemes that meet the above requirements have been developed and tested for a wide variety of problems. Two such schemes, will be discussed next.

V.4.1.1 Need for Flux Conservation

The most important requirement for the success of a patched grid scheme is that it be conservative in terms of computed fluxes. Before explaining how flux conservation can be achieved numerically, it is necessary to examine carefully why flux conservation is so important.

For this, consider the conservation of mass or continuity equation written in integral form :

$$\frac{\partial}{\partial t}\iiint_{V} \rho \, dV + \iint_{S} \rho \, \mathbf{V} \cdot \mathbf{dS} = 0 \tag{4.1}$$

In words, the time rate of decrease of mass inside control volume V is equal to the net flow of mass out of control volume through surface S. Hence, in order to prevent the creation or destruction of the quantity ρ, it is essential that the quantity given by

$$\iint_{S} \rho \, \mathbf{V} \cdot \mathbf{dS}$$

(called the flux) is conserved across the zonal boundary. Failure to conserve fluxes leads to incorrect positioning of shocks and discontinuities as illustrated in Figures V.4.5 and V.4.6 (from [28]) for the Burgers' equation ($u_t + uu_x = 0$).

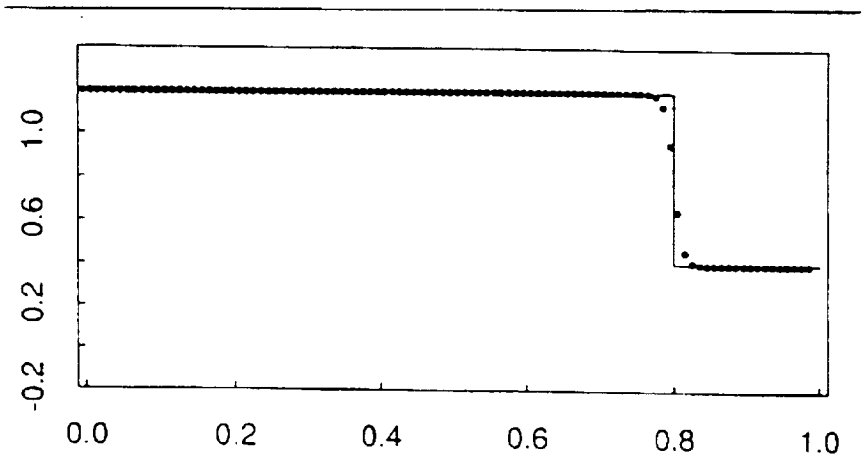


Figure V.4.5 : True and Computed Solutions to the Burgers' Equation using a Conservative Method

V.4.2 Giles' Approach for Patched Grid Calculations

To compute flows in a rotor-stator stage, Giles [19] uses a grid composed of two parts, one part fixed to the stator blade rows and the other part moving with the rotor with some prescribed velocity. The two parts are separated by a cell width at the interface with equal grid node spacing along

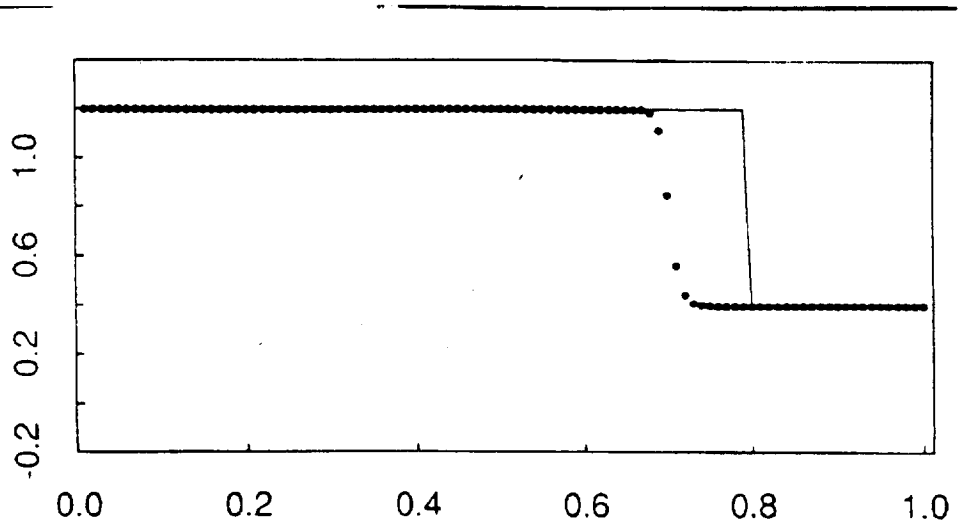


Figure V.4.6 : True and Computed Solutions to the Burgers' Equation using a Non-Conservative Method

the interface on either side. To span the gap between the two halves, a set of quadrilateral cells is defined by connecting each stator grid point to the nearest rotor grid node. As computations proceed in time, grid connectivity in amongst the quadrilateral cells changes dynamically as shown in Figure V.4.7 undergoing transformation from stage 1 through stage 2. As flow computations are for 2-dimensional cascades, assumptions are made regarding spatial periodicity of flow and hence the quadrilateral cells switch back to stage 1 after stage 2 is reached.

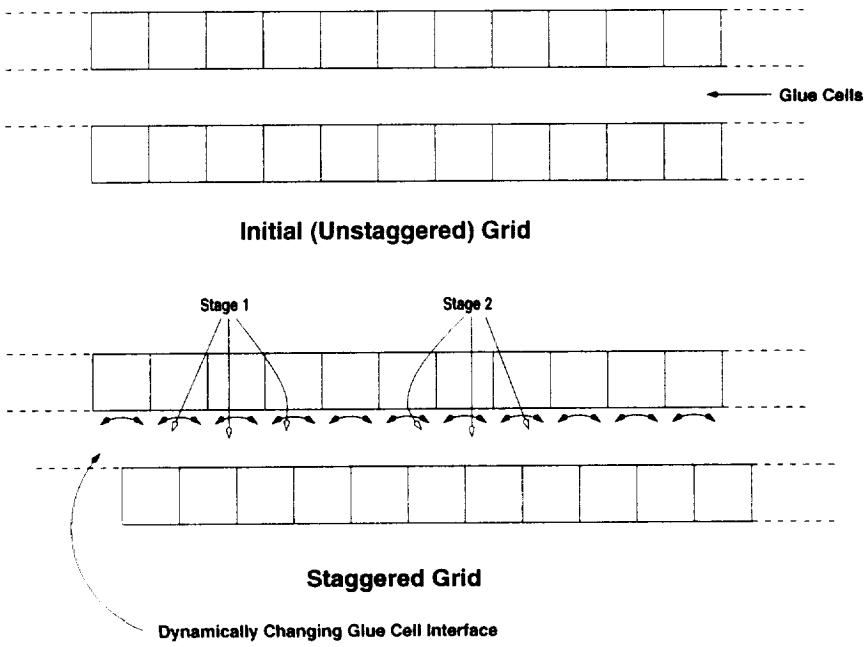


Figure V.4.7 : Giles' Approach for Euler Computations on Discontinuous Grids

This method presents the following problems for adaptations to 3-dimensional rotor-stator interaction simulations :

1. The developed method is for flows with spatial periodicity and hence cannot be readily used for more general simulations.
2. It uses the concept of remeshing as computations proceed. Such an approach in parallel 3-dimensional computations would be very expensive if at all possible.
3. In order to allow successful remeshing at intervals of time, equal grid node spacing is prescribed on either side of the interface. This may not be possible if unstructured grids are used for flow computations.
4. No details about the conservative nature of the numerical scheme have been provided.

The above drawbacks makes this method less suitable for 3-dimensional applications.

V.4.3 A Conservative Treatment for Zonal Grids

Rai [29, 34, 35, 36] developed a method in the mid 1980s for accurate and efficient computation of Euler flows using patched grids. The key feature of this approach was the emphasis laid on conservative treatment of zonal boundary conditions. Although this scheme was developed for finite-difference (FD) computations on structured grids, the underlying idea is not too difficult to apply to unstructured grids.

The salient steps of Rai's approach for a simple two-zone case can be briefly described as follows :

1. Select one of the zones for variable interpolation and the other for flux interpolation.
2. Extrapolate grid lines from the flux interpolation zone into the variable interpolation zone to generate extrapolated "ghost" cells.
3. Estimate the values of conserved variables at the vertices of the ghost cells.
4. Compute fluxes on the cells of the flux interpolation zone based on the ghost cells vertices' conserved variable values.
5. Obtain fluxes on the variable interpolation zone by conservative interpolation of fluxes from the flux interpolation zone.

This process will be explained with the help of the patched grid as shown in Figure V.4.8. For the sake of illustration, without any loss of generality, let zone 1 be the zone for flux interpolation and zone 2 be that for variable interpolation. To update the flow computation, it is essential to compute the fluxes at points like O, L and M which lie on the interface of the two zones in addition to computation of fluxes at all other points. Point O belongs to zone 1, while points L and M belong to zone 2.

For the purpose of estimating fluxes around all grid points, computational cells are constructed by joining the centers of all the quadrilaterals forming the grid. These cells are shown by dashed lines in Figure V.4.8. At zone 1 interface points such as point O, cells cannot be completed as the grid does not extend into the space beyond line AB. To enable cell generation, points R and S are located in zone 2 by extrapolation of grid lines from zone 1 to zone 2. Values of dependent

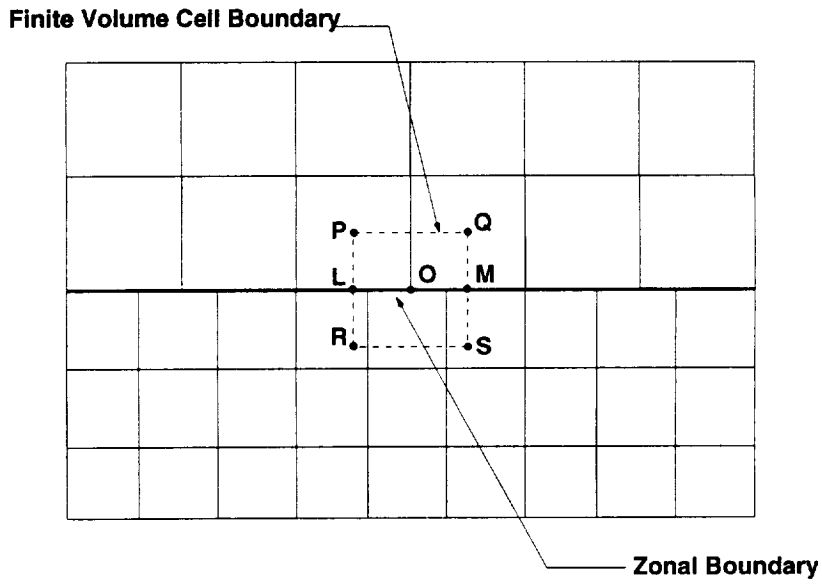


Figure V.4.8 : Grid Extrapolation for Rai's Conservative Zonal Scheme

variables at points R and S can be obtained by interpolation of dependent variables of zone 2. Once this is done, flux at point O can be computed using standard procedures. After computing all the fluxes at interface points like O on the side of zone 1, fluxes at points like L and M on the side of zone 2 can be obtained by using a conservative interpolation of fluxes of zone 1 interface points. Conservative interpolation means that the influx and efflux out of zone 1 should be balanced by the efflux and influx out of zone 2. Several conservative interpolation methods are possible, and Rai describes one such based on constant cell values for fluxes interpolation which can be found in [34] or [35].

V.4.4 Summary

The following is a summary of this section and serves to highlight the topics discussed :

1. The motivation behind investigation of rotor-stator interaction phenomena was explained and key issues for computational treatment pointed out.
2. A brief overview of existing methods for flow analysis was given with a particular emphasis on patched grid calculations.
3. Rai's conservative treatment for patched grid computations was explained.
4. Finally, the process of conservative interpolation and transfer between two meshes was explained in some detail.

V.5 Flow Computations using Patched Unstructured Grids

To the best of the writer's knowledge, all attempts at using patched grids for flow computations have been for finite-difference methods on structured (regular) grids. Patched unstructured grid computations have not been hitherto attempted because grid generation for complex geometries is relatively easier when unstructured grids are used. However, in case of relative motion between sub-grids, as in the case of rotor-stator interaction, it becomes essential to develop a methodology to enable flow computations to be performed on unstructured grids even when grid lines no longer remain continuous on account of mesh motion, see Figure V.4.1. This is one of the key objectives of current research.

This section begins with a description of an existing 2-dimensional CFD solver similar to the 3-dimensional solver used for aeroelastic computations described earlier Section V.3.3.1. Special attention will be given to the spatial discretization. Problems arising out of grid discontinuity for this type of spatial discretization along with a possible method for solution will be presented next. The chapter will conclude with some thoughts on conservative interpolation methods for such 2-dimensional computations.

V.5.1 A 2-Dimensional Unstructured Fluid Solver

A 2-dimensional Navier-Stokes [12, 25] solver using a mixed finite-volume formulation on unstructured triangular meshes is described here. For the case under study, namely discontinuous unstructured grids, the viscous terms have been neglected for the sake of simplicity and therefore description of the finite-element discretization of viscous terms will be omitted for brevity.

V.5.1.1 Governing Equations

Let $\Omega \subset \mathbb{R}^2$ be the flow domain of interest and Γ be its boundary. The conservative law form of the equations describing 2-dimensional Euler flows is given by :

$$\frac{\partial}{\partial t} W(\vec{x}, t) + \vec{\nabla} \cdot \vec{\mathcal{F}}(W(\vec{x}, t)) = 0 \tag{5.1}$$

where $\vec{x}$ and t denote the spatial and temporal variables, and

$$W = (\rho, \rho u, \rho v, E)^T , \qquad \vec{\nabla} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right)^T$$

and

$$\vec{\mathcal{F}} = \begin{pmatrix} F(W) \\ G(W) \end{pmatrix}$$

where $F(W)$ and $G(W)$ denote the convective fluxes given by :

$$F(W) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E + p) \end{pmatrix} , \qquad G(W) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E + p) \end{pmatrix}$$

In the above expressions, ρ is the density, $\vec{U} = (u, v)$ is the velocity vector, E is the total energy per unit volume and p is the pressure. The velocity, energy and pressure are related by the equation of state for a perfect gas :

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho\|\vec{U}\|^2\right)$$

where γ is the ratio of specific heats ($\gamma = 1.4$) for air.

V.5.1.2 Boundary Conditions

Three types of boundary conditions can be specified :

1. Inflow boundary condition : This is specified at the inlet for internal flow calculations.
2. Outflow boundary condition : This is specified at the exit for internal flow calculations.
3. Slip boundary condition : This is a no through-flow boundary condition to be imposed weakly at fixed walls.

V.5.1.3 Spatial Discretization

The flow domain Ω is assumed to be a polygonal bounded region of $\mathbb{R}^2$. Let $\mathcal{T}_h$ be a standard triangulation of Ω and h the maximal length of the edges of $\mathcal{T}_h$. The vertices of any triangle T are denoted by S_i and the set of its neighboring vertices by $K(i)$. A cell C_i for each vertex S_i is constructed as the union of the subtriangles resulting from the subdivision by means of the medians of each triangle of $\mathcal{T}_h$ that is connected to S_i Figure V.5.1. The boundary of C_i is denoted by ∂C_i and the unit outward normal to ∂C_i by $\vec{\nu} = (\nu_{ix}, \nu_{iy})$. The union of all these control volumes constitutes a discretization of domain Ω :

$$\Omega = \bigcup_{i=1}^{N_v} C_i$$

where N_v denotes the total number of triangle vertices in the grid. Figure V.5.2 indicates the dual finite-volume mesh associated with a typical unstructured triangulation.

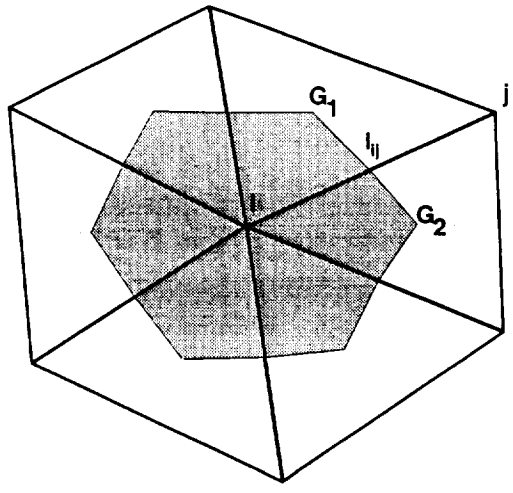


Figure V.5.1 : Cell Definition in an Unstructured Grid

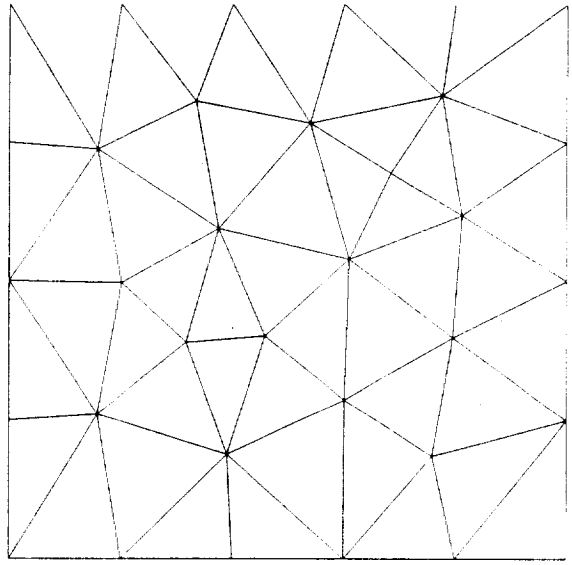


Figure V.5.2 : Dual Mesh Associated with a Typical Unstructured Triangulation

Integrate (5.1) over C_i to get

$$\frac{d}{dt}\int\int_{C_i} W\, d\vec{x} + \int\int_{C_i} \vec{\nabla}\cdot\vec{\mathcal{F}}(W)\, d\vec{x} = 0 \tag{5.2}$$

Integrate (5.2) by parts again to get

$$\begin{aligned}\frac{d}{dt}\int\int_{C_i} W\, d\vec{x} &+ \sum_{j\in K(i)}\int_{\partial C_{ij}} \vec{\mathcal{F}}(W)\cdot\vec{\nu}_{ij}\, d\sigma \qquad < 1 > \\ &+ \int_{\partial C_i\cap\Gamma_w} \vec{\mathcal{F}}(W)\cdot\vec{\nu}_i\, d\sigma \qquad < 2 > \\ &+ \int_{\partial C_i\cap\Gamma_{I/O}} \vec{\mathcal{F}}(W)\cdot\vec{\nu}_i\, d\sigma \qquad < 3 > \\ &= 0\end{aligned} \tag{5.3}$$

In the above expression Γ_b and $\Gamma_{I/O}$ are parts of the boundary of Ω at which the no-slip and the inlet/outlet boundary conditions are imposed such that $\Gamma = \Gamma_b \cap \Gamma_{I/O}$ and $\partial C_{ij} = \partial C_i \cap \partial C_j$

The above formulation leads to a locally one-dimensional computation for each convective term along the normal direction $\vec{\nu}$. For this purpose, the boundary ∂C_i of cell C_i is split into two bi-segments ∂C_{ij} which join the mid-point of the edge S_iS_j to the centroids of the triangles having both of S_i and S_j as vertices and the integral $< 1 >$ is evaluated as

$$\sum_{j\in K(i)}\int_{\partial C_{ij}} \vec{\mathcal{F}}(W)\cdot\vec{\nu}_{ij}\, d\sigma = \sum_{j\in K(i)} \vec{\mathcal{F}}(\tilde{U})\cdot\int_{\partial C_{ij}} \vec{\nu}_{ij}\, d\sigma \tag{5.4}$$

where $\vec{\mathcal{F}}(\tilde{U})$ is some approximation of the convective flux computed at the interface between cells C_i and C_j.

$\int_{\partial C_{ij}} \vec{\mathcal{F}}(W) \cdot \vec{\nu_{ij}}\, d\sigma$ is chosen to be a numerical flux function Φ associated with a first-order accurate upwind scheme [15]. It is denoted by $H_{ij}^{(1)}$ where the superscript (1) is used to indicated first-order accuracy. $H_{ij}^{(1)}$ can be written as :

$$H_{ij}^{(1)} = \Phi_{\mathcal{F}_{ij}} \left(W_i, W_j, \vec{n}_{ij} \right)$$

where $W_i = W(S_i)$ and $W_j = W(S_j)$ and

$$\vec{n}_{ij} = \int_{\partial C_{ij}} \vec{\nu}_{ij}\, d\sigma$$

As the Roe's [39] approximate flux function is used for computations, the expression for Φ becomes

$$\Phi_{\mathcal{F}}^{ROE}(U, V, \vec{n}) = \frac{\mathcal{F}(U, \vec{n}) + \mathcal{F}(V, \vec{n})}{2} - d(U, V, \vec{n})$$

where

$$\mathcal{F}(U, \vec{n}) = \vec{\mathcal{F}}(U) \cdot \vec{n}$$

$d(U, V, \vec{n})$ is the numerical artificial viscosity defined as :

$$d(U, V, \vec{\nu}) = \left| \mathcal{A}\left(\tilde{W}, \vec{n} \right) \right| \left(\frac{(V - U)}{2} \right)$$

and $\mathcal{A}$ is Roe's mean value of the flux Jacobian matrix $\dfrac{\partial \mathcal{F}}{\partial W}$.

V.5.1.4 Higher Order Extension

The numerical integration with an upwind scheme, as mentioned above, is only first-order accurate. A second order extension of Van Leer's MUSCL [26] method is developed for unstructured meshes for enhanced accuracy.

Based on the spatial approximation used in this method, the gradient of any function is constant over each cell of the mesh. Following the MUSCL method, one way to achieve second-order accuracy is to extrapolate the values of W_i and W_j at the cell interfaces $\partial C_i \cap \partial C_j$ to get W_{ij} and W_{ji} respectively given by

$$W_{ij} = W_i + \frac{1}{2} (\nabla W)_i^{\beta} \cdot \overrightarrow{S_i S_j}$$

$$W_{ji} = W_j - \frac{1}{2} (\nabla W)_j^{\beta} \cdot \overrightarrow{S_i S_j}$$

Here the approximate nodal gradients $(\nabla W)_{i,j}^{\beta}$ are obtained via a β-combination of centered and fully upwind gradients :

$$(\nabla W)_i^{\beta} = (1-\beta)\,(\nabla W)_i^{\text{CENT}} + \beta\,(\nabla W)_i^{\text{UPW}}$$

The centered gradient $(\nabla W)_i^{\text{CENT}} = (\nabla W)_{i,j}^{\beta=0}$ can be chosen as any vector satisfying

$$(\nabla W)_i^{\text{CENT}} \cdot \overrightarrow{S_i S_j} = W_j - W_i$$

To compute the upwind gradient, note that $(\nabla W)_i^{\text{UPW}} = (\nabla W)_{i,j}^{\beta=1}$. Then it follows that

$$(\nabla W)_i^{\text{UPW}} = 2(\nabla W)_i^{\beta=\frac{1}{2}} - (\nabla W)_i^{\text{CENT}}$$

The half upwind gradients ($\beta = 1/2$) are computed via a linear interpolation of the Galerkin gradients computed in each triangle of C_i so that

$$\begin{aligned}(\nabla W)_i^{\beta=1/2} &= \frac{\iint_{C_i} \nabla W|_{\triangle}\, dxdy}{\iint_{C_i} dxdy} \\ &= \frac{1}{\text{area}(C_i)} \sum_{\triangle \subset C_i} \frac{\text{area}(\triangle)}{3} \sum_{k=1, k\subset\triangle}^{3} W^k \nabla N_k^{\triangle}\end{aligned}$$

where $N_k^{\triangle}$ is the P1 shape function associated with node k of triangle $\triangle$. The final gradients are evaluated using a third-order biased scheme

$$\begin{aligned}(\nabla W)_i^{\beta=\frac{1}{3}} &= \frac{2}{3}(\nabla W)_i^{\beta=0} + \frac{1}{3}(\nabla W)_i^{\beta=1} \\ &= \frac{2}{3}(\nabla W)_i^{\beta=0} + \frac{1}{3}\left(2(\nabla W)_i^{\beta=\frac{1}{2}} - (\nabla W)_i^{\beta=0}\right) \\ &= \frac{1}{3}(\nabla W)_i^{\beta=0} + \frac{2}{3}(\nabla W)_i^{\beta=\frac{1}{2}}\end{aligned}$$

The flux in (5.4) is then taken to be $H_{ij}^{(2)}$ where the superscript (2) indicates second-order accuracy, given by :

$$H_{ij}^{(2)} = \Phi_{\mathcal{F}_{ij}}\left(W_{ij}, W_{ji}, \vec{n}_{ij}\right) \tag{5.5}$$

V.5.1.5 Implementation of Boundary Conditions

The terms $< 2 >$ and $< 3 >$ in (5.3) contain the physical boundary conditions. At the wall boundary, the slip condition ($\vec{U} \cdot \vec{\nu} = 0$) is imposed in the weak form and hence $< 2 >$ does not need to be evaluated. It can be verified that

$$\vec{U} \cdot \vec{\nu} = 0 \Longrightarrow \int_{\partial C_i \cap \partial \Gamma_w} \mathcal{F}(W, \vec{\nu}) d\sigma = p \begin{pmatrix} 0 \\ n_{i_{\Gamma_x}} \\ n_{i_{\Gamma_y}} \\ 0 \end{pmatrix} \quad \text{with} \quad \vec{n}_{i_\Gamma} = \int_{\partial C_i \cap \partial \Gamma_w} \vec{\nu}_{ij} d\sigma$$

At the inflow and outflow boundaries, a precise set of compatible exterior data values that depend on the flow regime and the velocity direction need to be specified. The integral $< 3 >$ is evaluated using a non-reflective version of the flux-splitting method of Steger and Warming [43] :

$$\int_{\partial C_i \cap \Gamma_{I/O}} \vec{\mathcal{F}}(W) \cdot \nu_i d\sigma = \mathcal{A}^+(W_i, \vec{n}_{I/O}) \cdot W_i + \mathcal{A}^-(W_i, \vec{n}_{I/O}) \cdot W_{I/O}$$

V.5.1.6 Time Discretization and Integration

The spatial discretizations explained above lead to the following semi-discrete fluid flow equations :

$$\frac{dW}{dt} + \psi(W) = 0 \tag{5.6}$$

A 3-step variant of the Runge-Kutta method is used for integrating the above equations. This may be summarized as :

$$\begin{cases} W^{(0)} = W^n \\ W^{(k)} = W^{(0)} - \dfrac{\Delta t}{4-k} \psi(W^{(k-1)}) \qquad k = 1, 2, 3 \\ W^{(n+1)} = W^{(3)} \end{cases}$$

This scheme can be shown to be third-order accurate for linear systems but only second-order accurate for non-linear equations such as the Euler equations.

V.5.2 Adaptation for Patched Unstructured Grids

To modify the existing 2-dimensional fluid solver for patched grid calculations, it was decided to extend Rai's approach outlined in Section V.4.3 to unstructured grids. Two methods need to be devised :

1. A compatible method for grid extrapolation retaining the same order of accuracy as the original unstructured code, and
2. A conservative interpolation method to transfer fluxes from one sub-mesh to another.

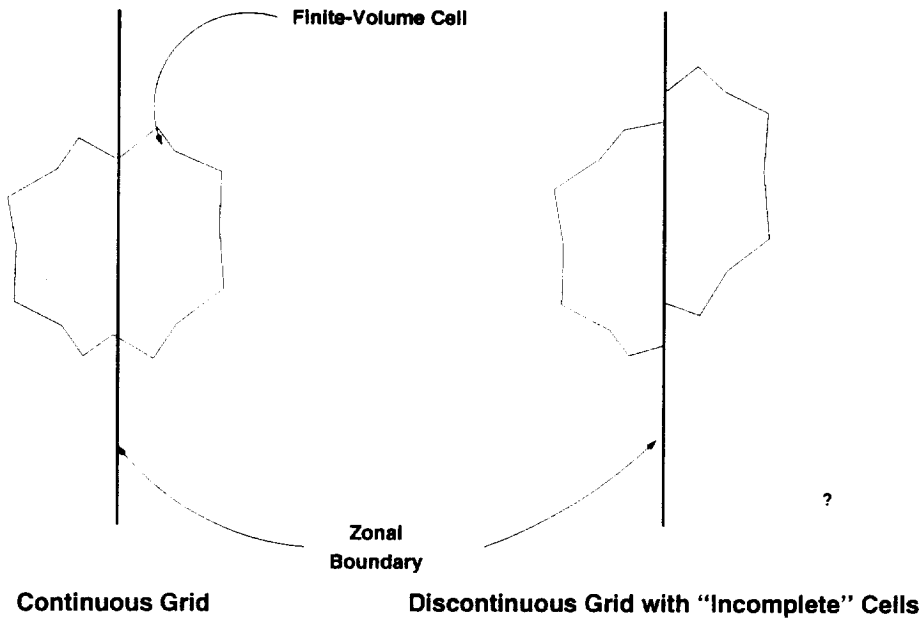


Figure V.5.3 : Continuous and Discontinuous Unstructured Grids

V.5.2.1 Grid Extrapolation

As there is no regular structure for unstructured grids, there is no obvious method for grid extrapolation for these unlike that for the case of structured meshes where grid lines are extended to generate new cells for interpolation. The main goal of this procedure was to generate "complete" cells for grid vertices on the sub-mesh interface at which fluxes are to be computed, see Figure V.5.3.

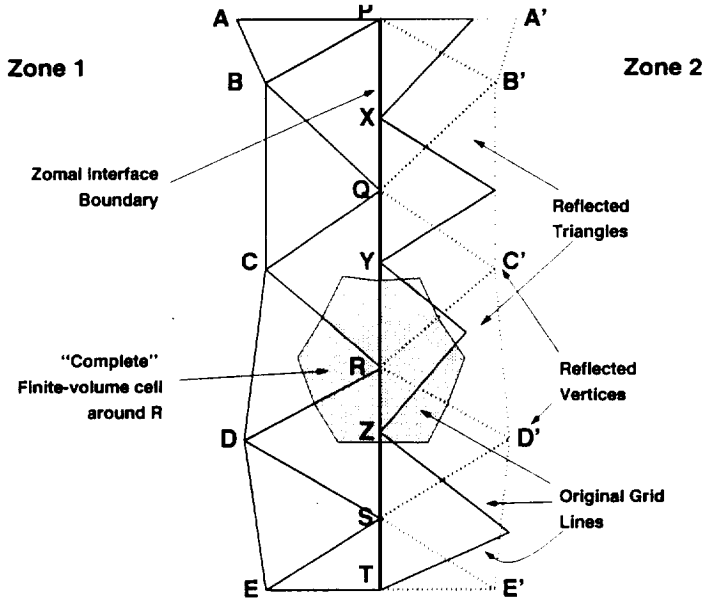


Figure V.5.4 : Grid Extrapolation & Cell Construction for Discontinuous Unstructured Grids

One way of doing this was to merely symmetrize the fluid grid triangles at the interface and generate new triangles penetrating into the other sub-mesh. This would involve the following steps :

1. For each triangle having an edge on the sub-mesh interface, project the triangle vertex not lying on the interface onto the other sub-mesh. This was a relatively simple operation as the interface between the two meshes was assumed to be a straight line (x = constant). Hence the co-ordinate of the new vertex can be easily obtained by retaining the y co-ordinate of the original vertex and considering the distance of the original vertex from the interface along the x-axis.
2. Generate connectivity information for the new triangles constructed. This uses the vertex numbers of the nodes of the original triangle lying on the interface and the newly assigned vertex number for the projected node.
3. Generate segment information for the segments of the new triangle. This is done using the same method as used in the original fluid code.

Once the projected triangles are generated, "complete" cell construction is straightforward as illustrated in Figure V.5.4. In this figure, the primed letters denoted the projected vertices from the original (unprimed) vertices of zone 1 into zone 2.

Another issue that needs to be handled is the determination of the triangles in which the newly created nodes lie. This is the traditional point-location problem from computer science for which many solutions have been proposed. Most of these solutions are based on a graphics and/or computational geometry point-of-view and require sophisticated and expensive algorithms. For the case of patched unstructured grids, the solution is greatly simplified on account of the fact that the region in which the point is to be located is divided into triangles. Having noted this, an idea is borrowed from triangular finite elements [14].

For linear interpolation of co-ordinates over triangles, three shape functions $N_{i,i=1,3}$ are defined such that

$$x = x_1 N_1 + x_2 N_2 + x_3 N_3$$
$$y = y_1 N_1 + y_2 N_2 + y_3 N_3$$

It is usual to define the shape functions N_i in terms of the parametric co-ordinated η_i. For triangles, the shape functions N_i are exactly identical to η_i. The parametric co-ordinates are defined such that $\sum_{i=1}^{3} \eta_i = 1$. Hence the following relationship is obtained in matrix form :

$$\begin{Bmatrix} 1 \\ x \\ y \end{Bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \begin{Bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \end{Bmatrix} \quad \text{or} \tag{5.7a}$$

$$\begin{Bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \end{Bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix}^{-1} \begin{Bmatrix} 1 \\ x \\ y \end{Bmatrix} \tag{5.7b}$$

Thus given any point with co-ordinates (x_p, y_p) and any triangle with co-ordinates $(x_i, y_i)_{i=1,3}$, one can find the parametric co-ordinates $\eta_{i,i=1,3}$ using relation (5.7b). Then whether or not the point lies inside the triangle can be easily determined by checking whether all the η_i values lie between 0 and 1. If all the values are between 0 and 1, then the point lies inside the triangles, else it is outside. The matrix inversion in (5.7b) can be symbolically computed and hard-coded in a program subroutine and hence points can be located efficiently.

Although the above-described process is computationally efficient, it should be noted that points need to be located at frequent intervals in a 3-dimensional aeroelastic computation when the grid associated with the rotating component will undergo rigid body rotations. Hence, in order to reduce computational costs further, the following strategy is adopted :

1. Only triangles (or tetrahedra) close to the mesh interface need be searched in.
2. To further restrict the search, it is performed only in the band of triangles lying a region such that their minimum and maximum x co-ordinates are not less than or greater than the minimum and maximum x co-ordinates of the newly created points.
3. Triangles are sorted based on their maximum x co-ordinates and a search is made only in the triangles whose maximum x co-ordinate is not greater than the x co-ordinate of the point being located.
4. Finally a binary search is employed to cut down the number of triangles searched by considering only those triangles such that the minimum and maximum x co-ordinates of the triangle bracket the x co-ordinate of the point.

Efficiency in sorting and searching is achieved by using fast algorithms [33, 40]. For sorting, Quicksort an $O(N \log_e N)$ algorithm, is used. The binary search algorithm is an $0(\log_e(N + 1))$ operation.

V.5.2.2 Flux and Variable Interpolation

Exchange of information between different sub-meshes is the crucial component of the patched grid approach for fluid computations. In order to allow free and undisturbed transition of shocks and discontinuities, it is essential that fluxes are interpolated conservatively. Several methods for flux interpolation were tried during the recent few months, details of which are explained below.

V.5.2.2.1 Conservative Interpolation of Fluxes based on Rai's Method

This is a straightforward extension of Rai's method to the case of unstructured grids and consists of the following steps :

1. "Complete" fluid-volume cells are constructed by symmetrization of triangles lying on the zonal interface.
2. Values of independent variables are obtained at the vertices of the newly created nodes by performing linear interpolation of variables at the vertices of the triangles in which they lie as shown in Figure V.5.4.
3. Once this is done, fluxes can be computed at the zonal interface points P, Q, R, S and T just as they would be for the case of continuous grids.
4. Fluxes computed at the above points are then interpolated conservatively to obtain fluxes at points X, Y and Z.
5. Independent variables at these points are updated using the fluxes obtained by interpolation.
6. Values of independent variables at points P, Q, R, S and T are obtained by interpolation from values at points X, Y and Z.

This process is repeated until the desired end of computations.

The major drawback with this method is that the areas for cells associated with points P, Q, R, S and T and those with points X, Y and Z are different. Since the term $\psi(W)$ in (5.6) incorporates the area of each cell, using the same fluxes on both sides of the interface leads to a loss of accuracy in space.

V.5.2.2.2 Independent Flux Computations on Either Side of the Zonal Interface

To overcome the problems with the interpolation of fluxes as outlined above, another approach was experimented with. In this case, symmetrization was carried out on each side of the zonal interface and fluxes were computed independently, obviating the need for interpolation, see Figure V.5.5.

Such an interpolation scheme is expected to satisfy all the necessary pre-requisites for correct patched grid computations as fluxes are fully conserved locally within each cell on either side of the zonal interface and hence along the interface as a whole. However, problems arise when two points coincide on the zonal interface on either side. In such a case, there is a chance that cells for each of the coincident points will be different and hence the fluxes and consequently the values of the independent variables. This would again lead to losses in accuracy.

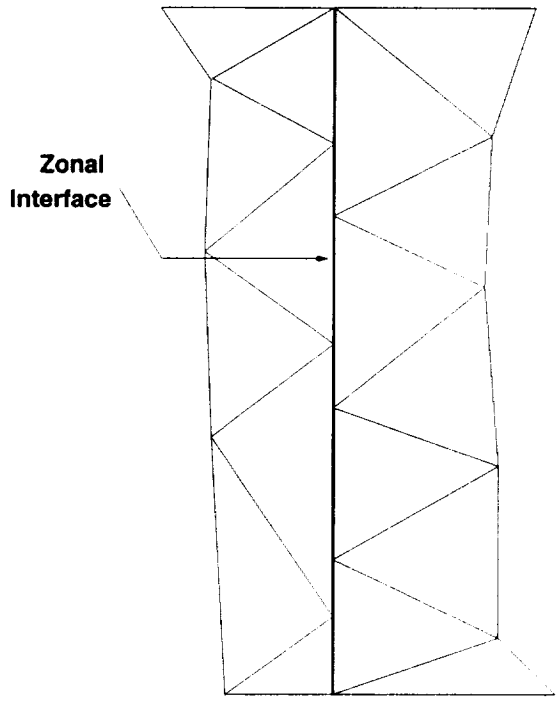


Figure V.5.5 : Double Symmetrization on the Zonal Interface

V.5.2.3 Consistent Flux Interpolation

The fundamental reason behind the losses of accuracy in the interpolation schemes mentioned above is the inconsistent manner in which fluxes are interpolated and transferred from one sub-mesh to another. In order to seek a remedy for these inconsistencies, one must first examine in detail the spatial discretization in the current scheme.

As outlined in Section V.5.1, fluxes at the vertices of each triangle are computed by solving the Riemann problem along each segment by which a vertex is connected to the other vertices of the mesh. The flux at the vertex is thus the sum of all fluxes thus obtained by solving the individual Riemann problems and is assumed to be *constant* throughout the finite volume cell as shown in Figure V.5.1. In the process of updating the flow solution as in (5.6), *both* the *area* and the *computed flux* for each are needed. Hence, for correct transfer of information, it is essential that both the correct areas and fluxes are used.

Currently, fluxes are interpolated across the zonal interface considering only one dimensional variation in the fluxes, that is, along the line of the zonal interface. Also, interpolation is carried out considering only the lengths of the segments at the zonal interface and no care is being taken to ensure that the proper areas are used for computation. The following interpolation scheme is suggested for consistent interpolation :

V.5.3 Suggested Interpolation Scheme

To develop a consistent interpolation scheme, the following points need first to be noted :

1. Fluxes are assumed constant throughout the entire finite-volume cell.

2. Flux interpolation has to be performed in two dimensions and not in one dimension.

   Assume that the sub-mesh from which fluxes are interpolated is extended into the other sub-mesh to a sufficient distance so that all cells for the vertices of that sub-mesh for which fluxes are to be interpolated are contained by the cells generated from the extension of the first sub-mesh.

   Consider a cell C belonging to the sub-mesh to which fluxes are to be interpolated as shown in Figure V.5.6. The boundary of C is shown in solid lines while those for cells belonging to the extended sub-mesh are shown in dotted lines. For the present, assume that the cell C overlaps the cells of the extended mesh so that it creates 4 areas of intersection, namely A_{i1}, A_{i2}, A_{i3} and A_{i4}.

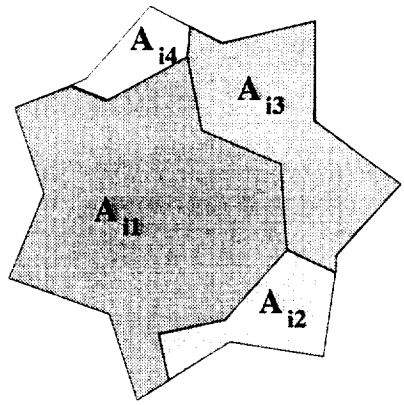


Figure V.5.6 : Intersection of Cells of Different Sub-Meshes for Flux Interpolation

Let the fluxes corresponding to these cells be F_1, F_2, F_3 and F_4 respectively and the whole areas of these cells be A_1, A_2, A_3 and A_4. Then, the flux contribution of the cell with area A_1 to cell C will be $(A_{i1}/A_1) \times F_1$ and likewise for the other cells. Thus, the flux for cell C will be given by,

$$F_c = \frac{A_{i1}}{A_1} F_1 + \frac{A_{i2}}{A_2} F_2 + \frac{A_{i3}}{A_3} F_3 + \frac{A_{i4}}{A_4} F_4 \tag{5.8}$$

Clearly, this will be conservative as the sum of all areas of intersection is equal to the area of cell C.

The interpolation method will thus comprise of the following steps :

1. Extend the grid to which fluxes are to be interpolated into the grid from which fluxes are to be interpolated to generate "complete" cells.
2. Determine the extent of the cells generated as mentioned above.
3. Extrapolate the grid from which fluxes are used for interpolation so that the cells of the extended grid "cover" the cells of the other sub-mesh lying on the interface generated in step 1.
4. Determine the extent of overlap of each cells of the extended mesh with cells generated in step 1.
5. At each time step, (a) compute fluxes on the sub-mesh from which fluxes are used for interpolation, (b) interpolate the fluxes to the other sub-mesh conservatively as in (5.8), (c) update the independent variables on this sub-mesh, and, (d) interpolate the independent variables back to the other sub-mesh.

The resulting scheme will be fully conservative. Also, as the only assumptions made are consistent with the assumptions made in the original Euler solver, namely, that variables vary linearly over the triangles and that fluxes are constant over each finite-volume cell, it is expected to create no additional errors on account of interpolation.

V.6 Results

This section describes the results obtained for discontinuous unstructured grids using the methods outlined in Section V.5.2.3 and mentions the salient points for the completion of proposed research.

V.6.1 Numerical Results

The shock tube problem proposed by Sod [41], provides a good test for the ability of a fluid solver to treat shocks and other discontinuities, is used often in analysis of methods in computational fluid dynamics. As the main focus of attention in developing a methodology to perform patched grid computations on unstructured grids revolves around allowing free and undisturbed passage of shocks and discontinuities, the shock problem has been used to test the method(s) for preliminary analysis. This section gives a brief overview of the physics of the shock-tube problem.

V.6.1.1 The Physical Shock Tube Problem

The shock tube is designed to trace the development of shocks and other discontinuities from a contact discontinuity in the initial state for the Euler equations given by (5.1).

The shock tube is a 1×1 (in physical dimensions), tube closed at both ends with a diaphragm separating a region of high-pressure (p_4) gas on the left from a region of low-pressure (p_1) gas on the right. This setup and initial state is illustrated in Figure V.6.1.

When the diaphragm is broken, a shock wave propagates into section 1 and an expansion wave propagates into section 4. This is illustrated in Figure V.6.2. As the normal shock wave propagates to the right, it increases the pressure behind it in region 2 and induces a mass motion in that region. The contact surface (interface between the region of high and low pressure) moves to the right with the same velocity as that of the mass motion in region 2. The expansion wave propagates to the left, smoothly and continuously decreasing the pressure in region 4 to the lower value p_3 behind the expansion wave.

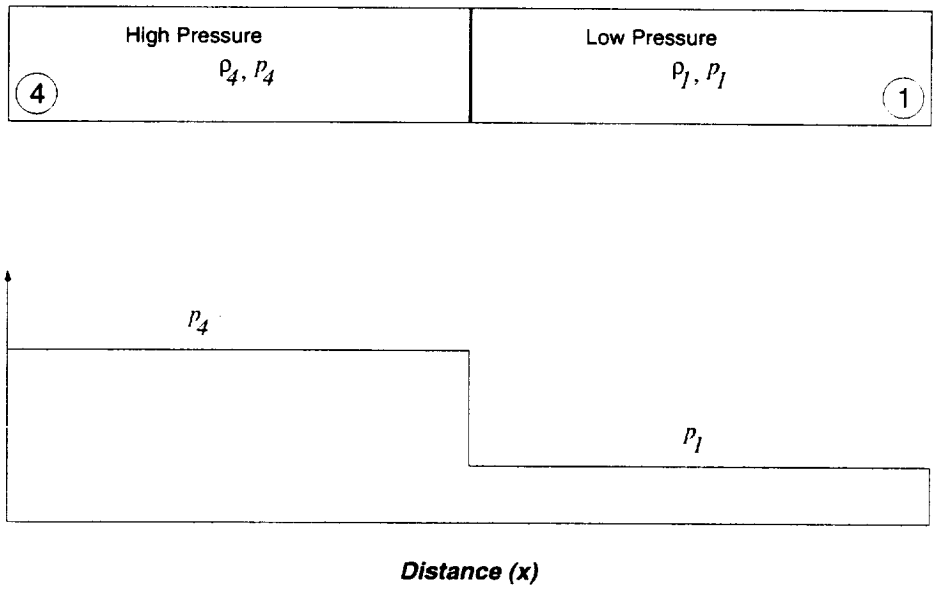


Figure V.6.1 : Initial State for the Shock Tube Problem

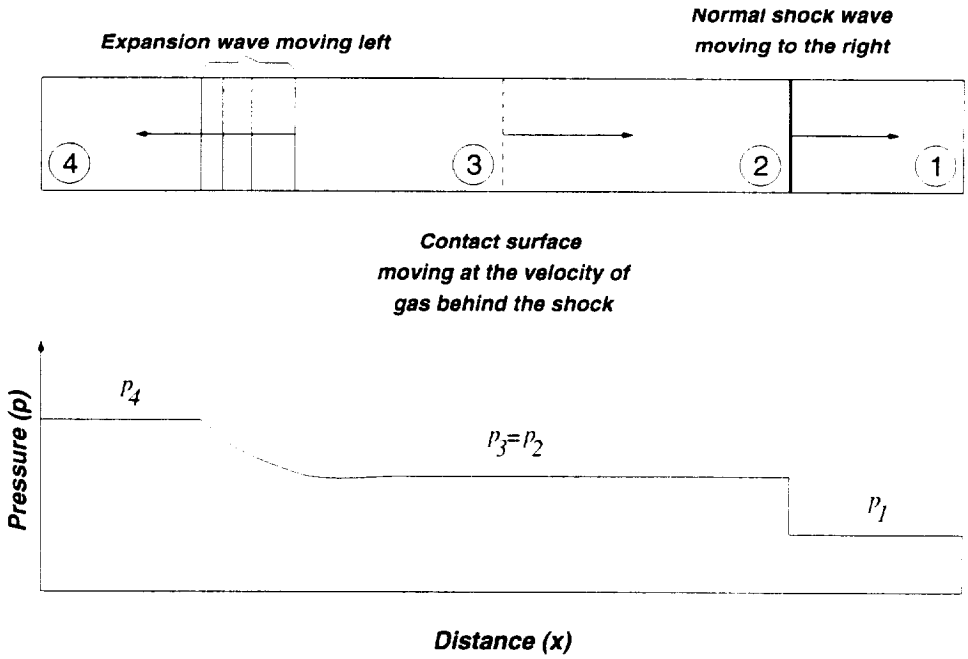


Figure V.6.2 : Flow in the Shock Tube after the Diaphragm is broken

For current simulations, the following initial conditions are used for the shock tube problem :

$$u_4 = u_1 = 0$$
$$p_4 = 1.0 \qquad p_1 = 0.1$$
$$\rho_4 = 1.0 \qquad \rho_1 = 0.125$$

V.6.1.2 Computational Setup for the Shock Tube Problem

Results for the current simulations are presented on 3 grids for the shock tube problems :

1. **Grid 1** : This is a patched grid made of two 3 × 51 grids which are reflections of each other. This grid has the special advantage that extrapolated "complete" cells are exactly identical and overlap each other completely as required by the suggested interpolation scheme presented in Section V.5.3. When first-order accuracy is used, interpolation of fluxes will be fully conservative and this will serve to illustrate the validity of the proposed method, see Figure V.6.3.
2. **Grid 2** : This is a more general patched grid and is used to illustrate the current status of the patched grid methodology and to highlight its features and drawbacks. It consists of two patched grids of 3 × 51 and 5 × 51, see Figure V.6.4.
3. **Grid 3** : Grids 1 and 2 mentioned above do not fully test the capabilities of the patched grid solver as the extent of required interpolation is limited. Grid 3 is therefore used to examine what happens when fluxes have to be interpolated over a larger zonal interface. This is a patched grid made of a 3 × 101 grid and 3 × 110 grid with the zonal interface being perpendicular to the contact discontinuity of the shock tube problem, see Figure V.6.5.

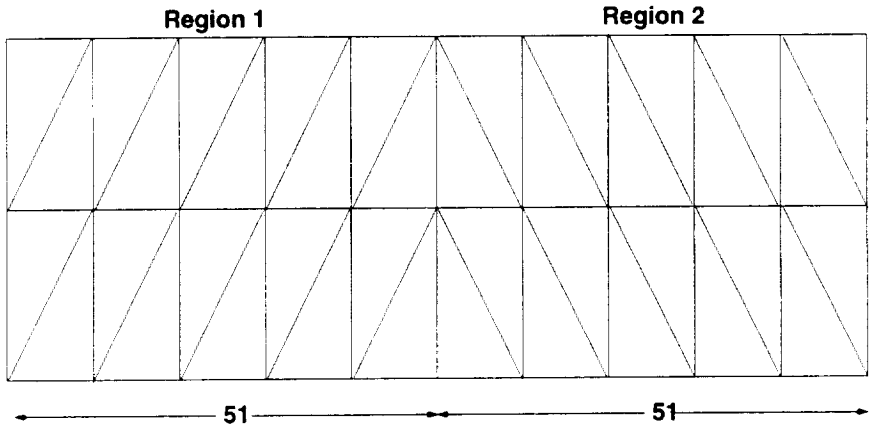


Figure V.6.3 : Grid 1 for Shock Tube Problem Computation

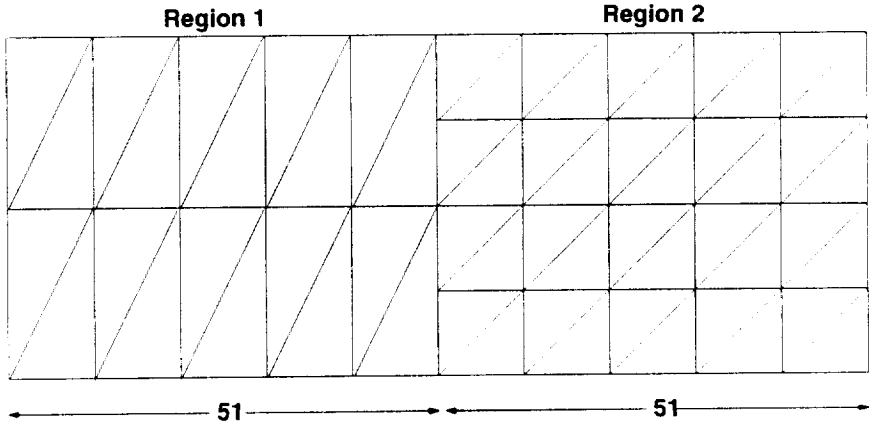


Figure V.6.4 : Grid 2 for Shock Tube Problem Computation

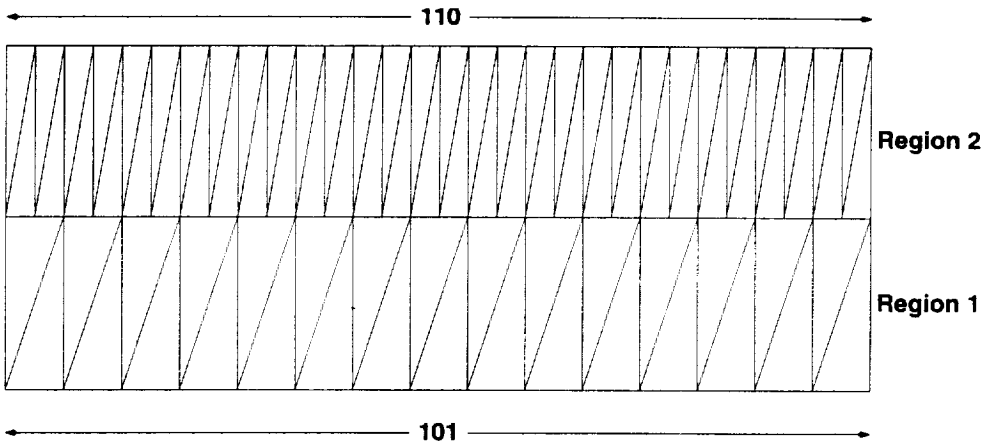


Figure V.6.5 : Grid 3 for Shock Tube Problem Computation

V.6.1.3 Results for the Shock Tube Problem

Typically, results for the shock tube presented at the physical time of $t = 0.16$ seconds. Two types of results are given, each with first- and second-order accuracy for both the grids mentioned above :

1. Flux interpolation at the interface based on Rai's method.
2. Independent flux computations by double symmetrization.

Each plot shown below is the variation of density with distance.

V.6.2 Conclusions

From the above experiments it can be see that :

1. Results for single and patched grid computations match fairly overall.
2. There is a loss in spatial accuracy near shocks and discontinuities when second-order accuracy is used which can be attributed inconsistent flux interpolation as mentioned in Section V.5.2.2.
3. For the case of first-order accuracy simulations for Grid 1, results from single and patched grids match exactly indicating the validity of the proposed solution method.

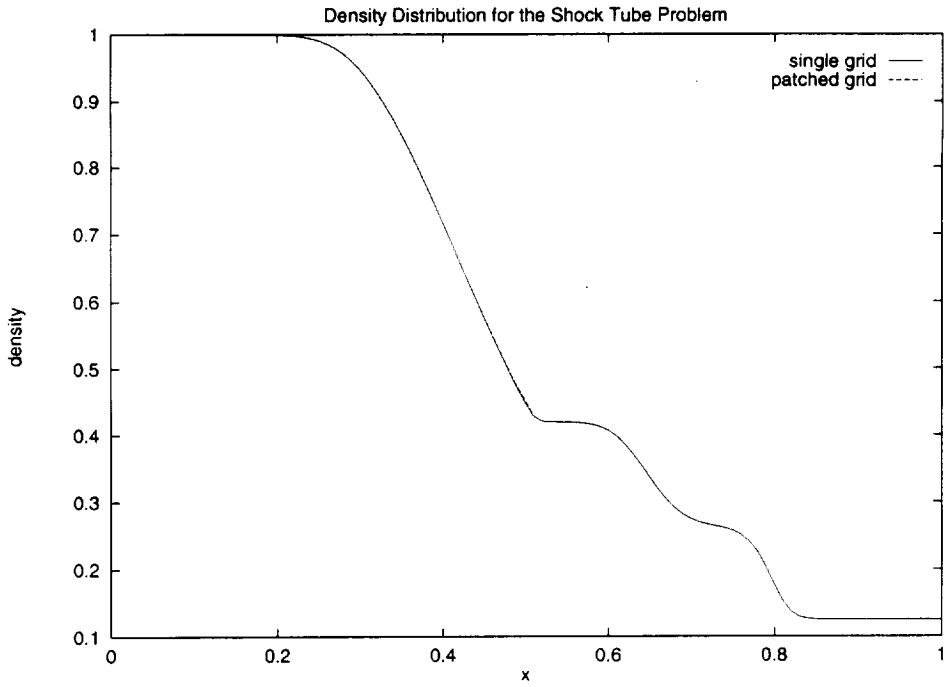


Figure V.6.6 : Results for Grid 1 with First-Order Accuracy using Flux Interpolation based on Rai's Method

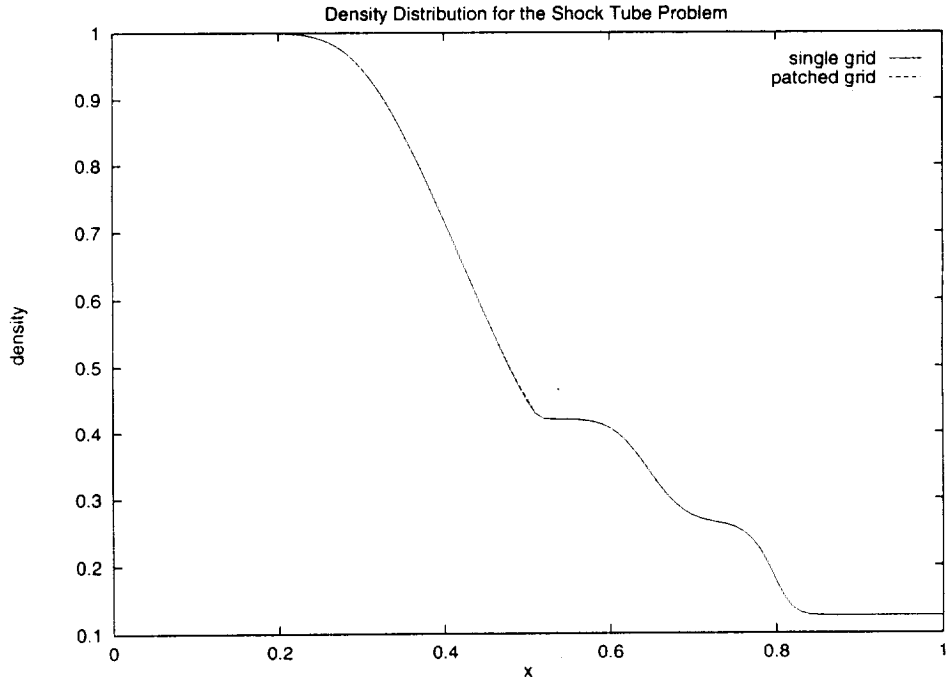


Figure V.6.7 : Results for Grid 1 with First-Order Accuracy with Independent Flux Computation

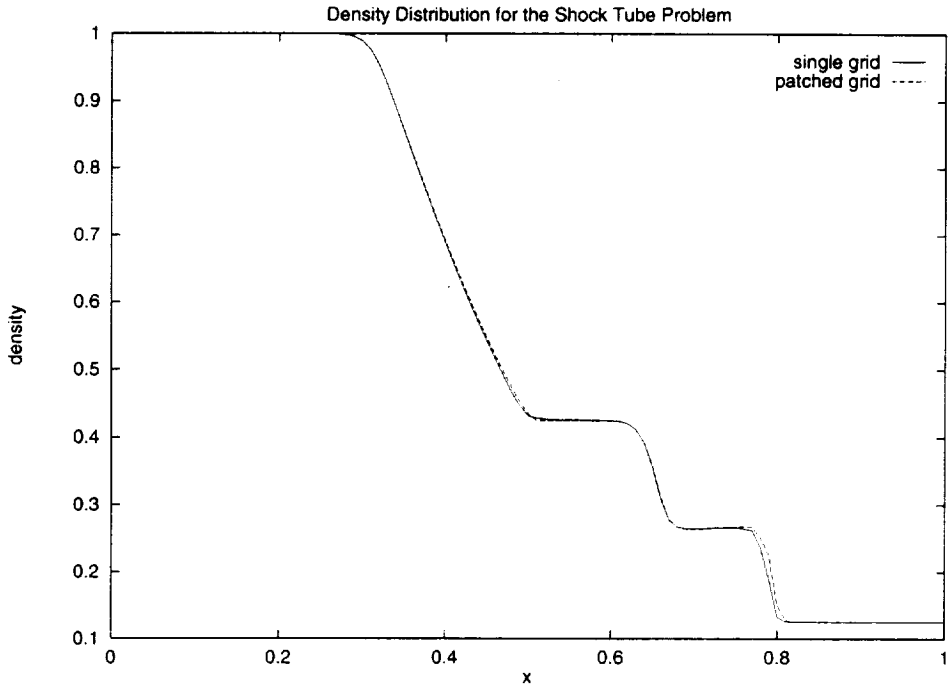


Figure V.6.8 : Results for Grid 1 with Second-Order Accuracy using Flux Interpolation based on Rai's Method

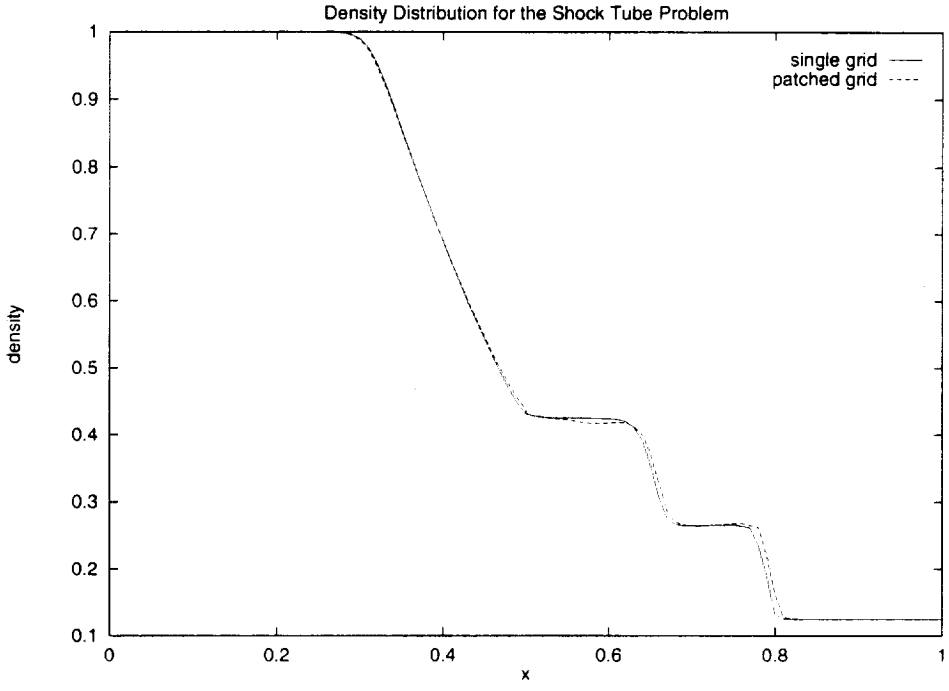


Figure V.6.9 : Results for Grid 1 with Second-Order Accuracy with Independent Flux Computation

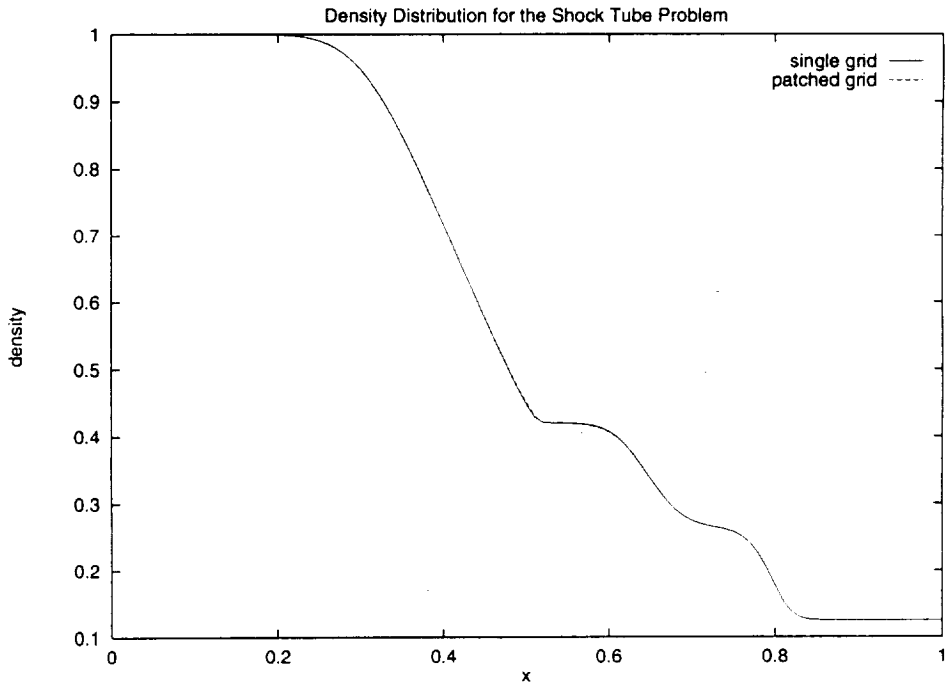


Figure V.6.10 : Results for Grid 2 with First-Order Accuracy using Flux Interpolation based on Rai's Method

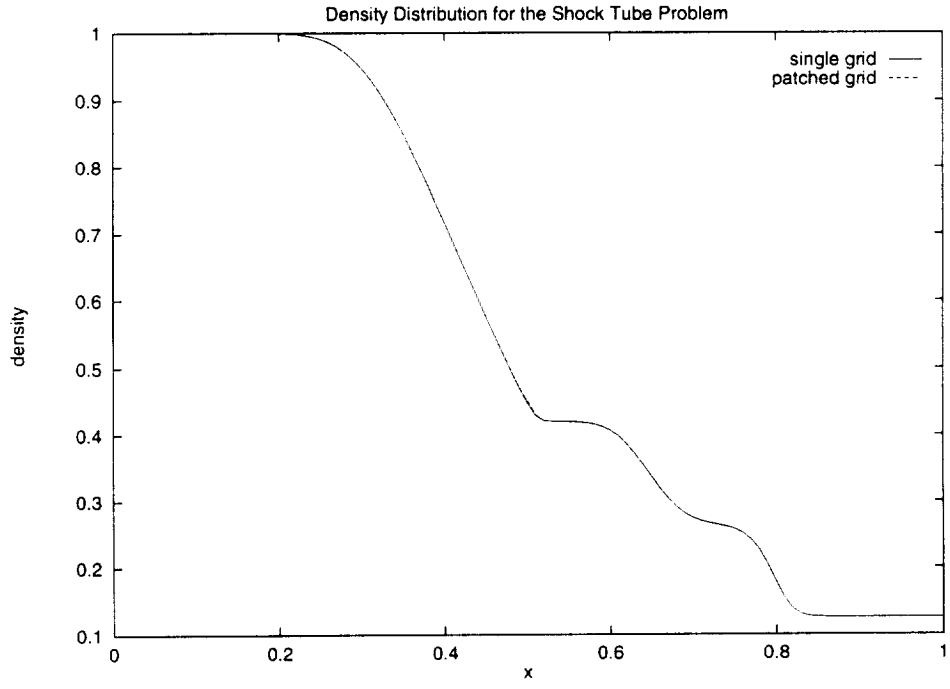


Figure V.6.11 : Results for Grid 2 with First-Order Accuracy with Independent Flux Computation

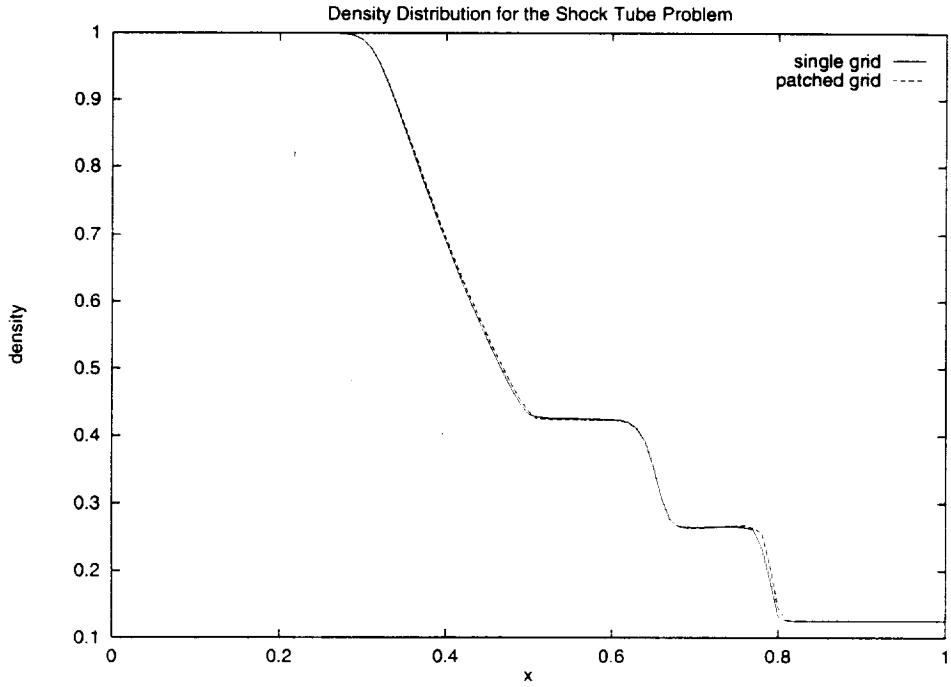


Figure V.6.12 : Results for Grid 2 with Second-Order Accuracy using Flux Interpolation based on Rai's Method

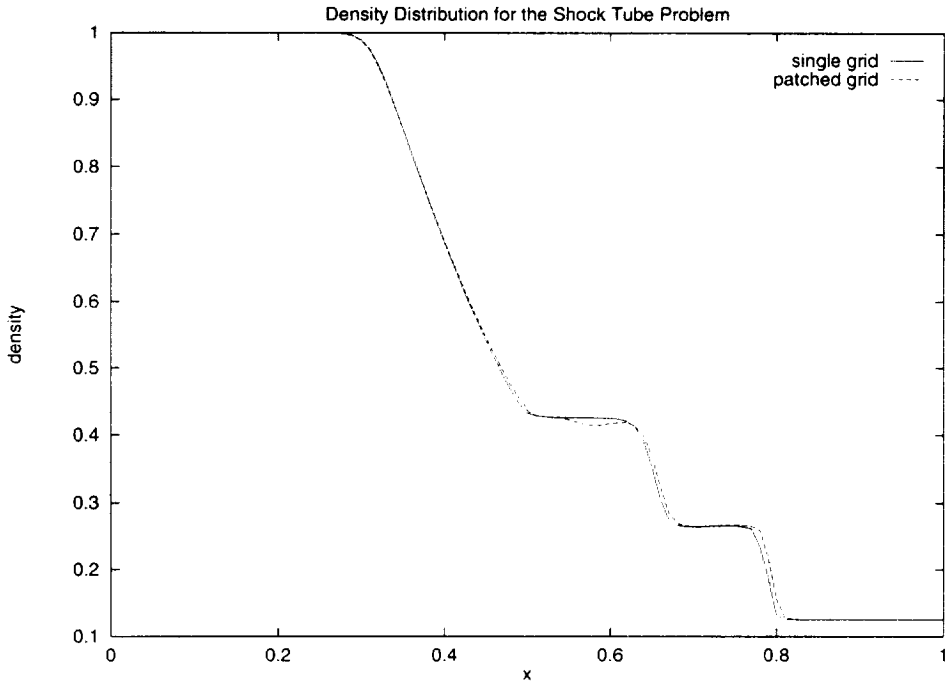


Figure V.6.13 : Results for Grid 2 with Second-Order Accuracy with Independent Flux Computation

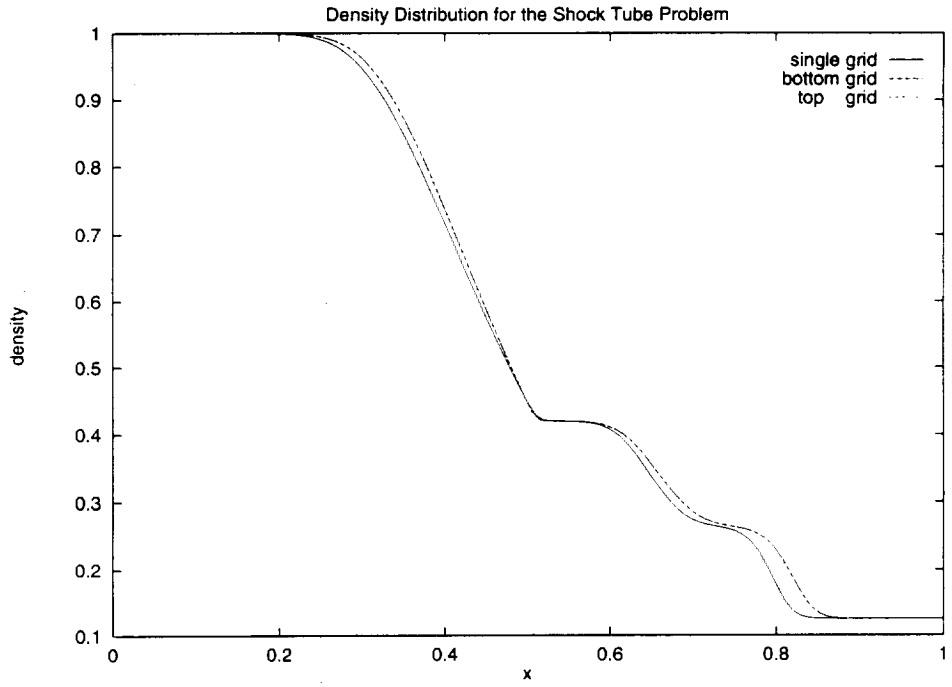


Figure V.6.14 : Results for Grid 3 with First-Order Accuracy using Flux Interpolation based on Rai's Method

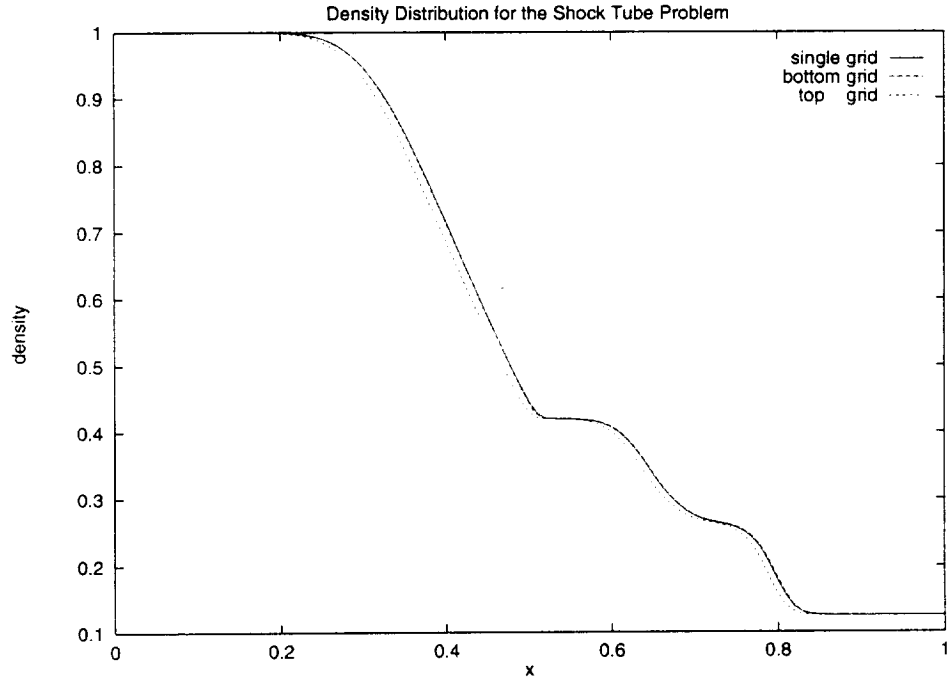


Figure V.6.15 : Results for Grid 3 with First-Order Accuracy with Independent Flux Computation

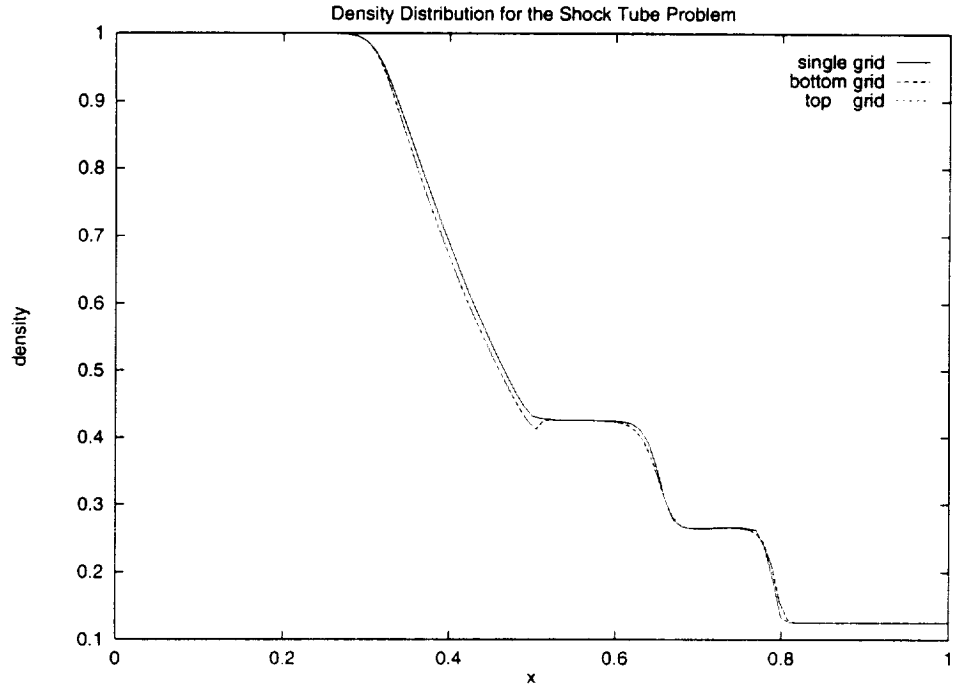


Figure V.6.16 : Results for Grid 3 with Second-Order Accuracy using Flux Interpolation based on Rai's Method

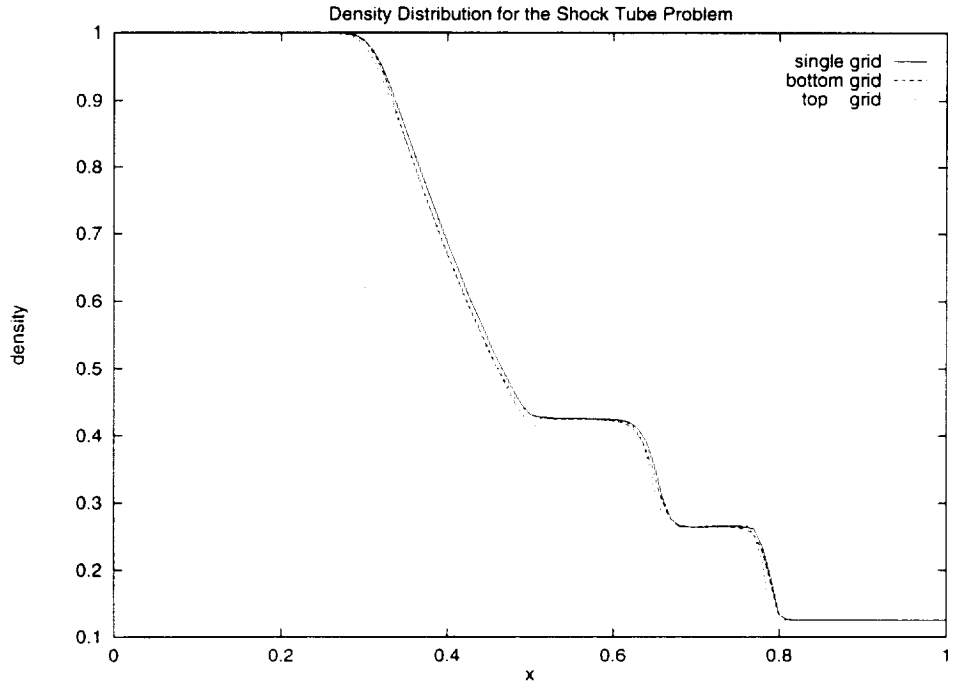


Figure V.6.17 : Results for Grid 3 with Second-Order Accuracy with Independent Flux Computation

V.7 Conclusions and Proposed Future Work

The main aim of research performed in the recent past was to develop a methodology for patched (discontinuous) unstructured grids. Emphasis was laid on the ability to transfer fluxes correctly so as to allow smooth and uninterrupted passage of shocks and other discontinuities.

A method was developed following the method of Rai [34] in which the grid of one zone is extrapolated into another and fluxes are computed by interpolating the independent variables at the extrapolated grid points. These fluxes are then interpolated conservatively to the other grid at which the solution is updated. The independent variables are then interpolated back to the extrapolated grid.

This method has been fairly successful in capturing shocks and discontinuities as has been illustrated by numerical experiments on the shock tube problem. This is especially true in the case when first-order accuracy is used in computation and the results obtained for a single grid and a patched grid match almost exactly, see Figures V.6.6, V.6.7 and V.6.10, V.6.11. For second-order spatial accuracy, however, a slight loss in spatial accuracy is observed near shocks although results in other regions of flow are in good agreement with those for a single grid, see Figures V.6.8, V.6.9 and V.6.12, V.6.13.

In terms of computational costs, the additional computations do not impose a severe burden on the efficiency of the fluid solver and it is expected that this cost would not be much compared to the time required for a fluid iteration.

A new method has been proposed which would attempt to alleviate the problems associated with losses in spatial accuracy and this has been confirmed with the results obtained for first-order accuracy in space for Grid 1 in Section V.6.1.3 (Figures V.6.6, V.6.7. To implement this method, an efficient algorithm to compute the intersection of two arbitrary simple polygons in 2-dimensions and two arbitrary polyhedra in 3-dimensions will be needed. A brief literature review of the area of computational geometry revealed the existence of many such algorithms and only the selection of an appropriate algorithm and its integration into the fluid code remain.

At present, only preliminary results on the shock tube problem have been given. In order to investigate the methods further, experiments will have to be carried out for more complicated situations. An important area of investigation would be the effect of moving meshes on the flux interpolation process. Considerable attention will have to be devoted to this in order to avoid the creation of any unnatural numerical shocks. First, it is expected that these investigations will be carried out in 2-dimensions and then extended to 3-dimensions after which truly unsteady simulations of rotor-stator interaction phenomena can be carried out.

Bibliography

[1] J. J. Adamczyk. Model Equation for Simulating Flows in Multistage Turbomachinery. ASME Paper 85-GT-226, 1985.

[2] M. A. Bakhle, T. S. R. Reddy, and T. G. K. Jr. Time Domain Flutter Analysis of Cascades Using a Full-Potential Solver. *AIAA Journal*, 30(1):163–170, January 1992.

[3] J. T. Batina. Unsteady Euler Airfoil Solutions using Unstructured Dynamic Meshes. AIAA Paper 89-0115.

[4] O. O. Bendiksen. Aeroelastic Problems in Turbomachines. AIAA Paper AIAA-90-1157-CP.

[5] O. O. Bendiksen. Role of Shocks in Transonic/Supersonic Compressor Rotor Flutter. *AIAA Journal*, 24:1179–1186, July 1986.

[6] R. V. Chima. Development of an Explicit Multigrid Algorithm for Quasi Three-Dimensional Flows in Turbomachinery. NASA TM-87128, January 1986.

[7] R. V. Chima and J. W. Yokota. Numerical Analysis of Three-Dimensional Viscous Internal Flows. *AIAA Journal*, 28(5):798–806, May 1990.

[8] J. Donea. Arbitrary Lagrangian-Eulerian Finite Element Methods. In T. Belytschko and T. J. R. Hughes, editors, *Computational Methods for Transient Analysis*, volume 1 of *Computational Methods in Mechanics*, chapter 10, pages 473–516. North-Holland, 1983.

[9] J. I. Erdos, E. Alzner, and W. McNally. Numerical Solution of Periodic Transonic Flow through a Fan Stage. *AIAA Journal*, 15(11):1559–1568, November 1977.

[10] C. Farhat, L. Crivelli, and F. X. Roux. A Transient FETI Methodology for Large-Scale Parallel Implicit Computations in Structural Mechanics. *International Journal of Numerical Methods in Engineering*, 37:1945–1975, 1994.

[11] C. Farhat, L. Crivelli, and F. X. Roux. Extending Substructure Based Iterative Solvers to Multiple Load and Repeated Analyses. *Computer Methods in Applied Mechanics and Engineering*, 1994:195–209, 1994.

[12] C. Farhat and S. Lanteri. Simulation of Compressible Viscous Flows on a variety of MPPs : Computational Algorithms for Unstructured Dynamic Meshes and Performance Results. *Computer Methods in Applied Mechanics and Engineering*, 119:35–60, 1994.

[13] C. Farhat, S. Lanteri, and H. D. Simon. TOP/DOMDEC — a Software Tool for Mesh Partitioning and Parallel Processing. *Computing Systems in Engineering*, 6(1):13–26, February 1995.

[14] C. A. Felippa. *Lecture Notes in Introduction to Linear Finite Element Methods*, volume II. University of Colorado, Boulder, 1989.

[15] L. Fezoui and B. Stoufflet. A Class of Implicit Upwind Schemes for Euler Simulations with Unstructured Meshes. *Journal of Computational Physics*, 84:174–206, 1989.

[16] S. Fleeter. Aeroelasticity Research for Turbomachine Applications. *Journal of Aircraft*, 16(5):320–326, May 1979.

[17] G. A. Gerolymos. Numerical Integration of the Blade-to-Blade Surface Euler Equations in Vibrating Cascades. *AIAA Journal*, 26(12):1483–1492, December 1988.

[18] G. A. Gerolymos. Advances in the Numerical Integration of Three-Dimensional Euler Equations in Vibrating Cascades. *Journal of Turbomachinery*, 115:781–790, October 1993.

[19] M. B. Giles. Stator/Rotor Interaction in a Transonic Turbine. *Journal of Propulsion and Power*, 6:621–627, Sept.–Oct. 1990.

[20] U. A. Gumaste, C. A. Felippa, and C. Farhat. Massively Parallel 3D Aeroelastic Analysis of Jet Engines. In *Computational Aerosciences Meeting*. NASA, 1996.

[21] J. L. Kerrebrock. *Aicraft Engines and Gas Turbines*. The MIT Press, 1992.

[22] M. Koya and S. Kotake. Numerical Analysis of Fully Three-Dimensional Periodic Flows through a Turbine Stage. *Journal of Engineering for Gas Turbines and Power*, 107:945–952, October 1985.

[23] F. Lane. Supersonic Flow Past an Oscillating Cascade with Supersonic Leading-Edge Locus. *Journal of the Aeronautical Sciences*, 24:65–66, January 1957.

[24] S. Lanteri. Parallel Solutions of Three-Dimensional Compressible Flows. Rapport de Recherche 2594, INRIA Sophia-Antipolis, June 1995.

[25] S. Lanteri and C. Farhat. Viscous Flow Computations on MPP systems : Implementational Issues and Performance Results for Unstructured Grids. In R. F. S. *et al.* editor, *Parallel Processing for Scientific Computing*, pages 65–70. SIAM, 1993.

[26] B. V. Leer. Towards the Ultimate Conservative Difference Scheme V : a Second-Order Sequel to Godunov's Method. *Journal of Computational Physics*, 32:361–370, 1979.

[27] M. Lesoinne. *Mathematical Analysis of Three-Field Numerical Methods for Aeroelastic Problems*. PhD thesis, University of Colorado, Boulder, 1994.

[28] R. J. LeVeque. *Numerical Methods for Conservation Laws*. Lectures in Mathematics - ETH Zürich. Birkhäuser Verlag, second edition, 1992.

[29] N. K. Madavan and M. M. Rai. Computational Analysis of Rotor-Stator Interaction in Turbomachinery Using Zonal Techniques. In P. A. Henne, editor, *Applied Computational Aerodynamics*, volume 125 of *Progress in Astronautics and Aeronautics*, chapter 13, pages 481–532. American Institute of Aeronautics and Astronautics, 1990.

[30] N. Maman and C. Farhat. Matching Fluid and Structure Meshes for Aeroelastic Computations : A Parallel Approach. *Computers and Structures*, 1994.

[31] K. C. Park and C. A. Felippa. Partitioned Analysis of Coupled Systems. In T. Belytschko and T. J. R. Hughes, editors, *Computational Methods for Transient Analysis*,

volume 1 of *Computational Methods in Mechanics*, chapter 10, pages 157–219. North-Holland, 1983.

[32] S. Piperno, C. Farhat, and B. Larrouturou. Partitioned procedures for the transient solution of coupled aeroelastic problems Part I : Model problem, theory and two-dimensional application. *Computer Methods in Applied Mechanics and Engineering*, 124:79–112, 1995.

[33] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in* FORTRAN — *The Art of Scientific Computing*. Cambridge University Press, second edition, 1992.

[34] M. M. Rai. A Conservative Treatment of Zonal Boundaries for Euler Equation Calculations. *Journal of Computational Physics*, 62:472–503, Feb. 1986.

[35] M. M. Rai. Navier-Stokes Simulations in Rotor/Stator Interaction using Patched and Overlaid Grids. *Journal of Propulsion and Power*, 3(5):387–396, Sept.-Oct. 1987.

[36] M. M. Rai and N. K. Madavan. Multi-Airfoil Navier-Stokes Simulations of Turbine Rotor-Stator Interaction. *Journal of Turbomachinery*, 112:377–384, July 1989.

[37] E. S. Reddy and C. C. Chamis. BLASIM : A Computational Tool to Assess Ice Impact on Engine Blades. AIAA Paper 93-1638.

[38] T. S. R. Reddy, M. A. Bakhle, R. Srivastava, O. Mehmed, and G. L. Stefko. A Review of Recent Aeroelastic Analysis Methods for Propulsion at NASA Lewis Research Center. NASA TP-3406, December 1993.

[39] P. L. Roe. Approximate Riemann Solvers, Parameter Vectors and Difference Schemes. *Journal of Computational Physics*, 43:357–371, 1981.

[40] R. Sedgewick. *Algorithms in C*. Addison-Wesley, 1990.

[41] G. A. Sod. A Survey of Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws. *Journal of Computational Physics*, 1978.

[42] R. Srivastava, L. N. Sankar, T. S. R. Reddy, and D. L. Huff. Application of an Efficient Hybrid Scheme for Aeroelastic Analysis of Advanced Propellers. *Journal of Propulsion*, 7(5):767–775, Sept–Oct 1991.

[43] J. L. Steger and R. F. Warming. Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite-Difference Methods. *Journal of Computational Physics*, 40:263–293, 1981.

[44] M. Stewart. Axisymmetric Aerodynamic Numerical Analysis of a Turbofan Engine. ASME Paper 95-GT-338, 1995.

[45] E. Sutjahjo and C. C. Chamis. Three-Dimensional Multidisciplinary Finite Elements for Coupled Analysis involving Fluid Mechanics, Heat Transfer and Solid Mechanics. AIAA Paper 96-1372.

[46] T. Tezduyar, M. Behr, and J. Liou. A New Strategy for Finite Element Computations involving Moving Boundaries and Interfaces — The Deforming Spatial Domain/Space-Time Procedure : I. The Concept and the Preliminary Numerical Tests. *Computer Methods in Applied Mechanics and Engineering*, 94:339–351, 1992.

[47] P. D. Thomas and C. K. Lombard. Geometric Conservation Law and Its Application to Flow Computations on Moving Grids. *AIAA Journal*, 17(10):1030–1037, October 1979.

[48] M. H. Vavra. *Aero-Thermodynamics and Flow in Turbomachines*. John Wiley & Sons, 1960.

[49] C.-H. Wu. A General Theory of Three-Dimensional Flow in Subsonic and Supersonic Turbomachines of Axial-, Radial-, and Mixed-Flow Types. NACA TN-2604, 1952.